INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

The Heart of Intersection Type Assignment

Steffen van Bakel

N° 00096419

Septembre 2006

Thème COM

ISSN 0249-6399 ISRN INRIA/RR--00096419--FR+ENG

The Heart of Intersection Type Assignment

Steffen van Bakel*

Thème COM — Systèmes communicants
Projets MIMOSA

Rapport de recherche n° 00096419 — Septembre 2006 — 34 pages

Abstract: This paper gives a new proof for the approximation theorem and the characterisation of normalisability using intersection types. The technique applied is to define reduction on derivations and to show a strong normalisation result for this reduction. From this result, the characterisation of strong normalisation and the approximation result will follow easily; the latter, in its turn, will lead to the characterisation of (head-)normalisability.

Key-words: intersection types, lambda calculus, normalisation, termination, cut-elimination.

* On sabbatical leave from Department of Computing, Imperial College London, 180 Queen's Gate London SW7 2BZ, U.K.

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : +33 4 92 38 77 77 — Télécopie : +33 4 92 38 77 65

Introduction

The Intersection Type Discipline as presented in [10] (a more enhanced system was presented in [9]; for an overview of the various existing systems, see [2]), is an extension of Curry's system [12], that consists mainly of allowing for term variables (and terms) to have more than one type. Intersection types are constructed by adding, next to the type constructor '$\rightarrow$' of Curry's system, the type constructor '$\cap$' and the type-constant 'ω'.

One way to explain the use of '$\cap$' is by stating that, in a certain context M, the term-variable x can play different, even non-unifyable, roles; this, in general, is a more liberal approach than just requiring only one type for x, and gives a greater expressiveness in terms of typeability. When more than one type has been assumed for x, then, by abstracting x over M, a term $\lambda x.M$ is obtained that accepts operands that have to satisfy more than just one requirement: a possible operand has to be typeable by each of the types used for x to type M. The type-constant 'ω' is the *universal type*, i.e. all terms can be assigned the type ω; this, in general, introduces non-normalisable, but typeable terms.

This slight generalisation causes a great change in complexity; in fact, now type assignment is closed for β-equality:

$$M =_\beta N \;\Rightarrow\; (B \vdash M{:}\sigma \iff B \vdash N{:}\sigma).$$

and (head / strong) normalisation can be characterized by assignable types:

$$\begin{array}{rcl} M \text{ has a head normal form} & \iff & B \vdash M{:}\sigma \;\&\; \sigma \neq \omega \\ M \text{ has a normal form} & \iff & B \vdash M{:}\sigma \;\&\; \omega \text{ does not occur in } B, \sigma \\ M \text{ is strongly normalisable} & \iff & B \vdash M{:}\sigma, \text{ where } \omega \text{ is not used at all.} \end{array}$$

(see, for example, [9, 1, 2]). These properties immediately show that type assignment (even in the system that does not contain ω, see [1]) is undecidable.

As in [22, 8], the set of terms can be extended by adding the term-constant $\bot$. Adding also the reduction rules $\bot N \rightarrow_{\beta\bot} \bot$, and $\lambda x.\bot \rightarrow_{\beta\bot} \bot$ to the notion of reduction gives rise to the notion of *approximate normal forms* that are in essense finite rooted segments of Böhm-trees. It is well known that interpreting a term by the set of approximants that can be associated to it, gives a model for the Lambda Calculus. From the Approximation Theorem, i.e. the observation that there exists a very precise relation between types assignable to a term M and those assignable to its approximants, $\mathcal{A}(M)$, formulated as

$$B \vdash M{:}\sigma \iff \exists\, A \in \mathcal{A}(M)\; [B \vdash A{:}\sigma]$$

(see [20, 1, 2]), it is immediately clear that the set of intersection types assignable to a term can be used to define a model for the Lambda Calculus (see [9, 1, 2]).

Of the above mentioned results, all but the first will be proved again in this paper; in fact, we will show that these can all be obtained from one more fundamental result.

In previous papers, the Approximation Theorem and Strong Normalisation Theorem were proved independently (see, respectively, [2] and [1]), though both using the same technique of Computability Predicates [21, 16]. This technique has been widely used to study normalisation properties or similar results, as for example in [11, 14, 19, 1, 2, 4, 5, 7, 6]. In this paper, we will show that both are special cases of a more fundamental result, using a variant of the technique developed in [7], that has also found its application in other fields [6]. This more fundamental result consists of defining a notion of reduction on derivations in '$\vdash$' that generalizes cut-elimination, and the proof of the theorem that this kind of reduction is strongly normalisable. It might seem surprising, but this result does not come easy at all. The reason for this is that, unlike for ordinary systems of type assignment, for the intersection system there is a significant difference between derivation reduction and ordinary reduction (see the beginning of Section *2.1*); unlike normal typed- or type assignment system, in '$\vdash$' not every term-redex occurs with types in a derivation. Moreover, especially the use of a relation '$\leq$' on types, together with a derivation rule $(\leq)$, greatly disturbs the smoothness of proofs (see again Section *2.1*).

From this strong normalisation result for derivation reduction, the Approximation Theorem and Strong Normalisation Theorem follow easily. The first of these implies the Head-Normalisation Theorem and (indirectly) the Normalisation Theorem, as was already demonstrated in [2].

The kind of intersection type assignment considered in this paper is that of [2], i.e. the essential intersection type assignment system, a restricted version of the BCD-system of [9], that is equally powerful in terms of typeability and expressiveness. The major feature of this restricted system is, compared to the BCD-system, a restricted version of the derivation rules and the use of strict types (first introduced in [1]).

In [3] a similar result was shown for the *strict* intersection type assignment system. This differs from the one considered here in that the $\leq$ relation on types used there is not contravariant over arrow types, but only allows for the selection of one of the types in an intersection. The contribution of this paper is to generalise that result to the *essential* intersection type assignment system, a notion of type assignment that is also closed for ν-reduction.

One of the first tentatives to tackle the main problem dealt with in this paper was to follow a very natural idea that originates from the observation formulated above: because of the presence of the type constant ω that can be assigned to any term, it is possible to type terms that contain non-normalizing subterms, so non-normalisable subterms are (at least partially) covered with ω. This lead to the assumption that, when defining a notion of $\perp$*-type assignment*, a variant of the essential system that consists basically of assigning

ω to the term-constant $\bot$ only, all typeable terms are strongly normalisable. This, perhaps surprisingly, turned out to be wrong, as will be illustrated in Section *5*.

The outline of this paper is as follows. In Section *1*, we will recall the definition of the essential type assignment system of [2], together with some of its main properties. In Section *2.1*, a notion of reduction on derivations in '$\vdash$' is defined, for which we will show a strong normalisation result in Section *2.2*. In section *5* we will present the $\bot$-system, a variant of the essential system that consists basically of assigning ω to the term-constant $\bot$ only. We will then show that, although in this system no redex can be covered by ω, this restriction itself is not enough to ensure strong-normalisability of typeable terms. However, in Section *5*, we will show that for the relevant intersection type assignment system [2], the restriction guarantees strong normalisation, and we will discuss the proof for the strong normalisation of derivation reduction in the strict system of [1]. We will finish this paper in Section *3.3* by extending the result of Section *2.2* to the characterisations of normalisation.

The result of this paper show that there *does* exist a relation between ω and redexes; in fact, in this paper we show that all the normalisation properties boil down to the same result: in a typeable term, all non-terminating subterms occur in (or are created by reduction in) positions that are typed with ω.

There exists a number of related results in the literature. For example, in [18] a strong normalisation result was proved for derivation reduction in the setting of the notion of intersection type assignment known as $\mathcal{D}$, as defined in [17]. This system is in fact the BCD-system [9] without the type-constant ω, and that strong normalisation result itself is a special case of the results of this paper presented in Section *5*.

Notations

In this paper, the symbol φ will be a type-variable; Greek symbols like $\alpha, \beta, \mu, \rho, \sigma$, and τ will range over types. '$\to$' will be assumed to associate to the right, and '$\cap$' binds stronger than '$\to$'. M, N are used for lambda terms, x, y, z for term-variables, $M[N/x]$ for the usual operation of substitution in terms, and A for terms in $\lambda\bot$-normal form. B is used for bases, and $B\backslash x$ for the basis obtained from B by erasing the statement that has x as subject. All symbols can appear indexed.

1 Intersection type assignment

In this section, the essential type assignment system of [2] is presented, a restricted version of the system presented in [9], together with some of its properties. The major feature of this restricted system is, compared to the BCD-system, a restricted version of the derivation

rules and the use of strict types. It also forms a slight extension of the strict type assignment system that was presented in [1]; the main difference is that the strict system is not closed for η-reduction, whereas the essential system is.

Definition 1.1 *i*) $\mathcal{T}_s$, the set of *strict types*, and $\mathcal{T}$, the set of *strict intersection types*, are defined through mutual induction by:

$$\begin{array}{rcl} \mathcal{T}_s & ::= & \varphi \mid (\mathcal{T} \to \mathcal{T}_s) \\ \mathcal{T} & ::= & (\mathcal{T}_s \cap \cdots \cap \mathcal{T}_s) \end{array}$$

ii) A *statement* is an expression of the form $M\!:\!\sigma$. M is the *subject* and σ the *predicate* of $M\!:\!\sigma$.

iii) A *basis* is a set of statements with only distinct variables as subjects.

iv) For bases $B_1, \ldots, B_n$, the basis $\cap\{B_1, \ldots, B_n\}$ is defined by: $x{:}\sigma_1 \cap \cdots \cap \sigma_m \in \cap\{B_1, \ldots, B_n\}$ if and only if $\{x{:}\sigma_1, \ldots, x{:}\sigma_m\}$ is the (non-empty) set of all statements about x that occur in $B_1 \cup \cdots \cup B_n$.

Notice that $\mathcal{T}_s$ is a proper subset of $\mathcal{T}$. Often $B, x{:}\sigma$ will be written for $\cap\{B, \{x{:}\sigma\}\}$, when x does not occur in B, and will omit the brackets '{' and '}'.

We define ω as the empty intersection: if $n = 0$, then $\sigma_1 \cap \cdots \cap \sigma_n \equiv \omega$, so ω does not occur in an intersection subtype. The motivation for this lies in the semantics of types (see [9]), where $[\![\sigma]\!]$ is the set of terms that can be assigned the type σ. Then, for all σ_i $(i \in \underline{n})$,

$$[\![\sigma_1 \cap \cdots \cap \sigma_n]\!] \subseteq [\![\sigma_1 \cap \cdots \cap \sigma_{n-1}]\!] \subseteq \ldots \subseteq [\![\sigma_1 \cap \sigma_2]\!] \subseteq [\![\sigma_1]\!].$$

It is natural to extend this sequence with $[\![\sigma_1]\!] \subseteq [\![\ ]\!]$, and therefore to define that the semantics of the empty intersection is the whole set of λ-terms, which is exactly $[\![\omega]\!]$.

Notice that intersection type schemes (so also ω) occur in strict types only as subtypes at the left-hand side of an arrow type scheme. Unless stated otherwise, if $\sigma_1 \cap \cdots \cap \sigma_n$ is used to denote a type, then all σ_i $(i \in \underline{n})$ are assumed to be strict.

Definition 1.2 (RELATIONS ON TYPES) *i*) The relation $\leq$ is defined as the least pre-order (i.e. reflexive and transitive relation) on $\mathcal{T}$ such that:

$$\begin{array}{rcl} \forall\, n \geq 1, \forall\, i \in \underline{n}\ [\sigma_1 \cap \cdots \cap \sigma_n & \leq & \sigma_i] \\ \forall\, n \geq 1, \forall\, i \in \underline{n}\ [\sigma \leq \sigma_i] & \Rightarrow & \sigma \leq \sigma_1 \cap \cdots \cap \sigma_n \\ \rho \leq \sigma\ \&\ \tau \leq \mu & \Rightarrow & \sigma \to \tau \leq \rho \to \mu \end{array}$$

ii) The equivalence relation $\sim$ on types is defined by: $\sigma \sim \tau \iff \sigma \leq \tau \leq \sigma$, and we will work with types modulo $\sim$.

iii) We write $B \leq B'$ if and only if for every $x{:}\sigma' \in B'$ there is an $x{:}\sigma \in B$ such that $\sigma \leq \sigma'$, and $B \sim B' \iff B \leq B' \leq B$.

Notice that $\mathcal{T}$ may be considered modulo $\sim$; then $\leq$ becomes a partial order. In this paper, however, in order to get a strong relation between the structure of types and derivations, types will not be considered modulo $\sim$.

The following property is easy to show:

Property 1.3 ([2]) *For all $\sigma, \tau \in \mathcal{T}$, $\sigma \leq \tau$ if and only if there are σ_i $(i \in \underline{n}), \tau_j$ $(j \in \underline{m})$ such that $\sigma = \sigma_1 \cap \cdots \cap \sigma_n$, $\tau = \tau_1 \cap \cdots \cap \tau_m$, and, for every $j \in \underline{m}$, there is an $i \in \underline{n}$ such that $\sigma_i \leq \tau_j$.* ■

The (essential) intersection type assignment system is constructed from the set of strict types and the following derivation rules. In this way a syntax directed system is obtained, that satisfies the main properties of the BCD-system (see [2]; the presentation of the derivation rules in that paper differs from that one used here).

Definition 1.4 *i*) *Intersection type assignment* and *intersection derivations* are defined by the following natural deduction system (where all types displayed are strict, except σ in the derivation rules $(\to I)$, $(\to E)$, and (Ax)):

$$(Ax): \frac{}{B, x{:}\sigma \vdash x{:}\tau}\,(\sigma \leq \tau) \qquad (\cap I): \frac{B \vdash M{:}\sigma_1 \quad \cdots \quad B \vdash M{:}\sigma_n}{B \vdash M{:}\sigma_1 \cap \cdots \cap \sigma_n}\,(n \geq 0)$$

$$(\to I): \frac{B, x{:}\sigma \vdash M{:}\tau}{B \vdash \lambda x.M{:}\sigma \to \tau} \qquad (\to E): \frac{B \vdash M{:}\sigma \to \tau \quad B \vdash N{:}\sigma}{B \vdash MN{:}\tau}$$

ii) We write $B \vdash M{:}\sigma$ if this statement is derivable using an intersection derivation, and write $\mathcal{D} :: B \vdash M{:}\sigma$ to specify that this result was obtained through the derivation $\mathcal{D}$.

Notice that $B \vdash M{:}\omega$, for all B and M, as a special case of rule $(\cap I)$.

We should emphasise the difference between this notion of type assignment and the strict one that was defined in [3]; instead of the rule (Ax) given above, it contained the rule

$$(\cap E): \frac{}{B, x{:}\sigma_1 \cap \cdots \cap \sigma_n \vdash_{\mathrm{s}} x{:}\sigma_i}\,(n \geq 1, i \in \underline{n})$$

Notice, that this rule is a special case of rule (Ax) in that $\sigma_1 \cap \cdots \cap \sigma_n \leq \sigma_i$, for all $i \in \underline{n}$. This is, in fact, the only difference between *strict* and *non-strict* type assignment. As

for the difference in derivable statements, in the essential system it is possible to derive $\vdash_\perp \lambda x.x : (\alpha{\to}\beta){\to}(\alpha{\cap}\gamma){\to}\beta$, which is not possible in '$\vdash_{\mathrm{S}}$'.

Some of the properties of this system, proved in [2], are:

Property 1.5 *i) If $B \vdash M : \sigma$, and $B' \leq B, \sigma \leq \tau$, then $B' \vdash M : \tau$, so the following rule is admissible in '$\vdash$':*

$$(\leq) : \frac{B \vdash M : \sigma}{B' \vdash M : \tau}\,(B' \leq B, \sigma \leq \tau)$$

In fact, if this rule is added to the system, the rule (Ax) can be replaced by:

$$(Ax) : \frac{}{B, x{:}\sigma \vdash x : \sigma}$$

ii) If $M \to_\eta N$, then $B \vdash M : \sigma$ if and only if $B \vdash N : \sigma$, so the following rule is admissible in '$\vdash$':

$$(\eta) : \frac{B \vdash M : \sigma}{B \vdash N : \sigma}\,(M \to_\eta N)$$

iii) If $M =_\beta N$, then $B \vdash M : \sigma$ if and only if $B \vdash N : \sigma$, so the following rule is admissible in '$\vdash$':

$$(=_\beta) : \frac{B \vdash M : \sigma}{B \vdash N : \sigma}\,(M =_\beta N)$$

We will use the following short-hand notation for derivations.

Definition 1.6 *i)* $\mathcal{D} = \langle Ax \rangle :: B \vdash x : \sigma$ if $\mathcal{D}$ consists of nothing but an application of rule (Ax).

ii) $\mathcal{D} = \langle \mathcal{D}_1, \ldots, \mathcal{D}_n, \cap I \rangle$, if and only if $\mathcal{D} :: B \vdash M : \sigma_1 {\cap} \cdots {\cap} \sigma_n$ for some σ_i $(i \in \underline{n})$, and there are derivations $\mathcal{D}_i :: B \vdash M : \sigma_i$ such that $\mathcal{D}$ is obtained from $\mathcal{D}_1, \ldots, \mathcal{D}_n$ by applying rule $(\cap I)$.

iii) $\mathcal{D} = \langle \mathcal{D}_1, {\to}I \rangle$, if and only if there are M_1, α, β such that $\mathcal{D} :: B \vdash \lambda x.M_1 : \alpha{\to}\beta$, and there is a derivation $\mathcal{D}_1 :: B, x{:}\alpha \vdash M_1 : \beta$, such that $\mathcal{D}$ is obtained from $\mathcal{D}_1$ by applying rule $({\to}I)$.

iv) $\mathcal{D} = \langle \mathcal{D}_1, \mathcal{D}_2, {\to}E \rangle$, if and only if there are P, Q, and τ such that $\mathcal{D} :: B \vdash PQ : \sigma$ and there are derivations $\mathcal{D}_1 :: B \vdash P : \tau{\to}\sigma$ and $\mathcal{D}_2 :: B \vdash Q : \tau$, such that $\mathcal{D}$ is obtained from $\mathcal{D}_1$ and $\mathcal{D}_2$ by applying rule $({\to}E)$.

We will identify derivations that have the same structure in that they have the same rules applied in the same order (so are derivations involving the same term); the types derived need not be the same.

We now extend the relation $\leq$ to derivations in $\vdash$; this notion is pivotal in the proof of strong normalisation of derivation reduction.

Definition 1.7 *i)* $\langle Ax\rangle :: B \vdash x:\sigma \leq \langle Ax\rangle :: B' \vdash x:\sigma'$ for all $B' \leq B$, and $\sigma \leq \sigma'$.

ii) $\langle \mathcal{D}_1, \ldots, \mathcal{D}_n, \cap I\rangle :: B \vdash M:\sigma_1 \cap \cdots \cap \sigma_n \leq \langle \mathcal{D}'_1, \ldots, \mathcal{D}'_m, \cap I\rangle :: B' \vdash M:\sigma'_1 \cap \cdots \cap \sigma'_m$, if and only if for every $j \in \underline{m}$ there exists a $i \in \underline{n}$ such that $\mathcal{D}_i \leq \mathcal{D}'_j$.

iii) $\langle \mathcal{D}_1 :: B, x{:}\alpha \vdash M:\beta, \rightarrow I\rangle :: B \vdash \lambda x.M:\alpha\rightarrow\beta \leq \langle \mathcal{D}'_1 :: B', x{:}\alpha \vdash M:\beta', \rightarrow I\rangle :: B' \vdash \lambda x.M':\alpha'\rightarrow\beta'$ if and only if $\mathcal{D}_1 \leq \mathcal{D}'_1$.

iv) Let $\mathcal{D} = \langle \mathcal{D}_1 :: B \vdash P:\tau\rightarrow\sigma, \mathcal{D}_2 :: B \vdash Q:\tau, \rightarrow E\rangle :: B \vdash PQ:\sigma$. Then, for every $\rho \leq \tau, \mu \geq \sigma$, $\mathcal{D}'_1 \leq \mathcal{D}_1$, $\mathcal{D}'_2 \geq \mathcal{D}_2$ such that $\mathcal{D}'_1 :: B' \vdash P:\rho\rightarrow\mu$ and $\mathcal{D}'_2 :: B' \vdash Q:\rho$,

$$\mathcal{D} \leq \langle \mathcal{D}'_1 :: B' \vdash P:\rho\rightarrow\mu, \mathcal{D}'_2 :: B' \vdash Q:\rho, \rightarrow E\rangle :: B' \vdash PQ:\mu.$$

Notice that $\leq$ is contra-variant in $(\rightarrow E)$.

The following is easy to show, and establishes the relation between $\leq$ on types and $\leq$ on derivations:

Lemma 1.8 *i) If $\mathcal{D} :: B \vdash M:\sigma$ and $B' \leq B$, $\sigma \leq \sigma'$, then there exists $\mathcal{D}' \geq \mathcal{D}$ such that $\mathcal{D}' :: B' \vdash M':\sigma'$.*

ii) If $\mathcal{D} :: B \vdash M:\sigma \leq \mathcal{D}' :: B' \vdash M':\sigma'$, then $B' \leq B$, $\sigma \leq \sigma'$.

Proof: *i*) We separate two cases:

($\sigma' \in \mathcal{T}_s$) : By induction on the structure of derivations.

(Ax) : Then $\mathcal{D} = \langle Ax\rangle :: B, x{:}\rho \vdash x:\sigma$, with $\rho \leq \sigma$. Since $B' \leq B, x{:}\rho$, there exists $x{:}\mu \in B'$ such that $\mu \leq \rho \leq \sigma \leq \sigma'$. Take $\mathcal{D}' = \langle Ax\rangle :: B' \vdash x:\sigma'$, then $\mathcal{D} \leq \mathcal{D}'$.

($\cap I$) : Then $\mathcal{D} = \langle \mathcal{D}_1, \ldots, \mathcal{D}_n, \cap I\rangle :: B \vdash M:\sigma_1 \cap \cdots \cap \sigma_n$, with $\mathcal{D}_i :: B \vdash M:\sigma_i$, for $i \in \underline{n}$; notice that $\mathcal{D} \leq \mathcal{D}_i$. Then, by Property *1.3*, there exists $j \in \underline{n}$ such that $\sigma_j \leq \sigma'$, and, by induction, there exists $\mathcal{D}'_j :: B' \vdash M:\sigma'$, with $\mathcal{D}_j \leq \mathcal{D}'_j$. Take $\mathcal{D}' = \mathcal{D}'_j$, then $\mathcal{D} \leq \mathcal{D}'$.

($\rightarrow I$) : Then $\mathcal{D} = \langle \mathcal{D}_1 :: B, x{:}\alpha \vdash M':\beta, \rightarrow I\rangle :: B \vdash \lambda x.M':\alpha\rightarrow\beta$, so $\sigma = \alpha\rightarrow\beta$. Since $\sigma' \in \mathcal{T}_s$, $\sigma' = \rho\rightarrow\mu$ such that $\rho \leq \alpha$ and $\beta \leq \mu$. Then $B', x{:}\rho \leq B, x{:}\alpha$, and by induction, there exists $\mathcal{D}'_1 \geq \mathcal{D}_1$, such that $\mathcal{D}'_1 :: B', x{:}\rho \vdash M':\mu$. Take $\mathcal{D}' = \langle \mathcal{D}'_1 :: B', x{:}\rho \vdash M':\mu, \rightarrow I\rangle :: B' \vdash \lambda x.M':\rho\rightarrow\mu$, then $\mathcal{D} \leq \mathcal{D}'$.

($\rightarrow$*E*) : Then $\mathcal{D} = \langle \mathcal{D}_1 :: B \vdash M_1 : \gamma \rightarrow \sigma, \mathcal{D}_2 :: B \vdash M_2 : \gamma, \rightarrow E\rangle :: B \vdash M_1 M_2 : \sigma$. Since $\gamma \rightarrow \sigma \leq \gamma \rightarrow \sigma'$, by induction, there exists $\mathcal{D}'_1 :: B' \vdash M'_1 : \gamma \rightarrow \sigma'$ such that $\mathcal{D}'_1 \geq \mathcal{D}_1$; notice that $\mathcal{D}_2 \leq \mathcal{D}_2$. Take $\mathcal{D}' = \langle \mathcal{D}'_1, \mathcal{D}_2, \rightarrow E\rangle :: B' \vdash M_1 M_2 : \sigma'$, then $\mathcal{D} \leq \mathcal{D}'$.

($\sigma' = \sigma'_1 \cap \cdots \cap \sigma'_n$) : By Property *1.3*, for $i \in \underline{n}$, $\sigma \leq \sigma'_i \in \mathcal{T}_s$; by part *(i)*, there exists $\mathcal{D}'_i \geq \mathcal{D}_i$ such that $\mathcal{D}'_i :: B' \vdash M : \sigma'_i$. Take $\mathcal{D}' = \langle \mathcal{D}'_i, \ldots, \mathcal{D}'_n, \cap I\rangle :: B' \vdash M : \sigma'$, then $\mathcal{D} \leq \mathcal{D}'$.

ii) Easy, from Definition *1.7*. ■

2 Strong normalisation of derivation reduction

In this section, we will define a notion of reduction on derivations and show this notion to be strongly normalisable.

2.1 Derivation reduction

In this section, we will define a notion of reduction on derivations $\mathcal{D} :: B \vdash M : \sigma$. This will follow ordinary reduction, by contracting typed redexes that occur in $\mathcal{D}$, i.e. redexes for subterms of M of the shape $(\lambda x.P)Q$, for which the following is a subderivation of $\mathcal{D}$:

$$\langle\langle \mathcal{D}_1 :: B, x{:}\sigma \vdash P : \tau, \rightarrow I\rangle :: B_1 \vdash \lambda x.P : \sigma \rightarrow \tau, \mathcal{D}_2 :: B \vdash Q : \sigma, \rightarrow E\rangle :: B \vdash (\lambda x.P)Q : \tau.$$

We will prove in Section *2.2* that this notion of reduction is terminating, i.e. strongly normalisable.

The effect of this reduction will be that the derivation for the redex $(\lambda x.P)Q$ will be replaced by a derivation for the contractum; this can be regarded as a generalisation of cut-elimination, but has, because the system at hand uses intersection types together with the relation '$\leq$', to be defined with care. Take for example the following derivation for $B \vdash (\lambda x.x)N : \sigma$.

$$\dfrac{\dfrac{}{x{:}\sigma \vdash x : \sigma}}{\vdash \lambda x.x : \sigma \cap \tau \rightarrow \sigma} \qquad \dfrac{\mathcal{D}_1 \quad \mathcal{D}_2}{B \vdash N : \sigma \qquad B \vdash N : \tau}$$

$$\dfrac{\vdash \lambda x.x : \sigma \cap \tau \rightarrow \sigma \qquad B \vdash N : \sigma \cap \tau}{B \vdash (\lambda x.x)N : \sigma}$$

This derivation will reduce to $\mathcal{D}_1 :: B_1 \vdash N : \sigma$; it is exactly the fact that the derivation $\mathcal{D}_2$ (and the derivation redexes that occur inside it) does not return in the contractum, that

makes this kind of reduction strongly normalizing. So, when contracting a derivation for the redex

$$\langle\langle \mathcal{D}_1 :: B, x{:}\sigma \vdash P{:}\tau, {\to}I\rangle :: B_1 \vdash \lambda x.P{:}\sigma{\to}\tau, \mathcal{D}_2 :: B \vdash Q{:}\sigma, {\to}E\rangle :: B \vdash (\lambda x.P)Q{:}\tau,$$

i.e.

$$\dfrac{\dfrac{\dfrac{}{x{:}\sigma \vdash x{:}\rho}\,(\sigma \le \rho)}{\mathcal{D}_1}}{\dfrac{B, x{:}\sigma \vdash P{:}\tau}{B \vdash \lambda x.P{:}\sigma{\to}\tau}} \qquad \dfrac{\mathcal{D}_2}{B \vdash Q{:}\sigma}$$
$$B \vdash (\lambda x.P)Q{:}\tau$$

it is in general not the case that the derivation $\mathcal{D}_2$ will just be inserted in the positions of $\mathcal{D}_1$ where a type for the variable x is derived, since that would give an illegal derivation. The $(\le)$-step 'to be applied at the end of $\mathcal{D}_2$' has to be pushed upwards; this is possible because of Property *1.5 (i)*. This, in general, changes the structure of the derivation.

Reduction on derivations is formally defined by first defining substitution on derivations:

Definition 2.1 (DERIVATION SUBSTITUTION) For $\mathcal{D} :: B, x{:}\sigma \vdash M{:}\tau$, and $\mathcal{D}_0 :: B \vdash N{:}\sigma$, the result $\mathcal{D}'$ of substituting $\mathcal{D}_0$ in $\mathcal{D}$, $\mathcal{D}\,[\mathcal{D}_0/x{:}\sigma] :: B \vdash M[N/x]{:}\tau$ is inductively defined by:

i) $\mathcal{D} = \langle Ax\rangle :: B, x{:}\sigma \vdash x{:}\tau$, with $\sigma \le \tau$. Let $\mathcal{D}'_0$ be such that $\mathcal{D}_0 \le \mathcal{D}'_0 :: B \vdash N{:}\tau$, then $\mathcal{D}\,[\mathcal{D}_0/x{:}\sigma] = \mathcal{D}'_0$.

ii) $\mathcal{D} = \langle \mathcal{D}_1, \ldots, \mathcal{D}_n, {\cap}I\rangle :: B, x{:}\sigma \vdash M{:}\tau_1{\cap}\cdots{\cap}\tau_n$, so for $i \in \underline{n}$, $\mathcal{D}_i :: B, x{:}\sigma \vdash M{:}\tau_i$. Let

$$\mathcal{D}'_i = \mathcal{D}_i\,[\mathcal{D}_0/x{:}\sigma] :: B \vdash M[N/x]{:}\tau_i,$$

then

$$\mathcal{D}' = \langle \mathcal{D}'_1, \ldots, D'_n, {\cap}I\rangle :: B \vdash M[N/x]{:}\tau_1{\cap}\cdots{\cap}\tau_n = \langle \mathcal{D}_1, \ldots, \mathcal{D}_n, {\cap}I\rangle[\mathcal{D}_0/x{:}\sigma].$$

iii) $\mathcal{D} = \langle \mathcal{D}_1 :: B, x{:}\sigma, y{:}\alpha \vdash M_1{:}\beta, {\to}I\rangle :: B, x{:}\sigma \vdash \lambda y.M_1{:}\alpha{\to}\beta$. Let

$$\mathcal{D}'_1 = \mathcal{D}_1\,[\mathcal{D}_0/x{:}\sigma] :: B, y{:}\alpha \vdash M_1[N/x]{:}\beta$$

Then

$$\mathcal{D}' = \langle \mathcal{D}'_1, {\to}I\rangle :: B \vdash (\lambda y.M_1)[N/x]{:}\alpha{\to}\beta = \langle \mathcal{D}'_1, {\to}I\rangle[\mathcal{D}_0/x{:}\sigma]$$

iv) $\mathcal{D} = \langle \mathcal{D}_1 :: B, x{:}\sigma \vdash P : \rho{\to}\tau, \mathcal{D}_2 :: B, x{:}\sigma \vdash Q : \rho, {\to}E\rangle :: B, x{:}\sigma \vdash PQ : \tau$. Let

$$\begin{array}{rcl} \mathcal{D}'_1 & = & \mathcal{D}_1\,[\mathcal{D}_0/x{:}\sigma] :: B \vdash P[N/x] : \rho{\to}\tau, \\ \mathcal{D}'_2 & = & \mathcal{D}_2\,[\mathcal{D}_0/x{:}\sigma] :: B \vdash Q[N/x] : \rho, \end{array}$$

then

$$\mathcal{D}' = \langle \mathcal{D}'_1, \mathcal{D}'_2, {\to}E\rangle :: B \vdash (PQ)[N/x] : \tau = \langle \mathcal{D}_1, \mathcal{D}_2, {\to}E\rangle[\mathcal{D}_0/x{:}\sigma]$$

Before coming to the definition of derivation-reduction, we need to define the concept of 'position of a subderivation in a derivation'.

Definition 2.2 Let $\mathcal{D}$ be a derivation, and $\mathcal{D}'$ be a subderivation of $\mathcal{D}$. The position p of $\mathcal{D}'$ in $\mathcal{D}$ is defined by:

i) If $\mathcal{D}' = \mathcal{D}$, then $p = \varepsilon$.
ii) If the position of $\mathcal{D}'$ in $\mathcal{D}_1$ is q, and $\mathcal{D} = \langle \mathcal{D}_1, {\to}I\rangle$, or $\mathcal{D} = \langle \mathcal{D}_1, \mathcal{D}_2, {\to}E\rangle$, then $p = 1q$.
iii) If the position of $\mathcal{D}'$ in $\mathcal{D}_2$ is q, and $\mathcal{D} = \langle \mathcal{D}_1, \mathcal{D}_2, {\to}E\rangle$, then $p = 2q$.
iv) If the position of $\mathcal{D}'$ in $\mathcal{D}_i$ ($i \in \underline{n}$) is q, and $\mathcal{D} = \langle \mathcal{D}_1, \ldots, \mathcal{D}_n, {\cap}I\rangle$, then $p = q$.

We now can give a clear definition of reductions on derivations; notice that this reduction corresponds to contracting a redex $(\lambda x.M)N$ in the term involved only if that redex appears in the derivation in a sub-derivation with type different from ω.

Definition 2.3 We say that the derivation $\mathcal{D} :: B \vdash M : \sigma$ *reduces to* $\mathcal{D}' :: B \vdash M' : \sigma$ *at position* p *with redex* R, if and only if:

i) $\sigma \in \mathcal{T}_s$.
a) $\mathcal{D} = \langle\langle \mathcal{D}_1, {\to}I\rangle, \mathcal{D}_2, {\to}E\rangle :: B \vdash (\lambda x.M)N : \sigma$

$$\dfrac{\dfrac{\begin{array}{c}\mathcal{D}_1 \\ B, x{:}\tau \vdash M : \sigma\end{array}}{B \vdash \lambda x.M : \tau{\to}\sigma} \quad \begin{array}{c}\mathcal{D}_2 \\ B \vdash N : \tau\end{array}}{B \vdash (\lambda x.M)N : \sigma}$$

Then $\mathcal{D}$ reduces to $\mathcal{D}_1\,[\mathcal{D}_2/x{:}\tau] :: B \vdash M[N/x] : \sigma$ at position ε with redex $(\lambda x.M)N$.
b) If $\mathcal{D}_1$ reduces to $\mathcal{D}'_1$ at position p with redex R, then
1) $\mathcal{D} = \langle \mathcal{D}_1, {\to}I\rangle :: B \vdash \lambda x.M_1 : \alpha{\to}\beta$ reduces to $\mathcal{D}' = \langle \mathcal{D}'_1, {\to}I\rangle :: B \vdash \lambda x.M_1 : \alpha{\to}\beta$ at position $1p$ with redex R.

2) $\mathcal{D} = \langle \mathcal{D}_1, \mathcal{D}_2, \rightarrow E\rangle :: B \vdash PQ : \sigma$ reduces to $\mathcal{D}' = \langle \mathcal{D}'_1, \mathcal{D}_2, \rightarrow E\rangle :: P'Q : \sigma$ at position $1p$ with redex R.

3) $\mathcal{D} = \langle \mathcal{D}_2, \mathcal{D}_1, \rightarrow E\rangle :: B \vdash PQ : \sigma$ reduces to $\mathcal{D}' = \langle \mathcal{D}_2, \mathcal{D}'_1, \rightarrow E\rangle :: PQ' : \sigma$ at position $2p$ with redex R.

ii) $\sigma = \sigma_1 \cap \cdots \cap \sigma_n$. If $\mathcal{D} :: B \vdash M : \sigma_1 \cap \cdots \cap \sigma_n$, then, for every $i \in \underline{n}$, there is a $\mathcal{D}_i$, such that $\mathcal{D}_i :: B \vdash M : \sigma_i$, and $\mathcal{D} = \langle \mathcal{D}_1, \ldots, \mathcal{D}_n, \cap I\rangle$. If there is an $i \in \underline{n}$ such that $\mathcal{D}_i$ reduces to $\mathcal{D}'_i$ at position p with redex $(\lambda x.P)Q$ (a subterm of M), then, for all $1 \leq j \neq i \leq n$, either

a) there is no redex at position p because there is no subderivation at that position, and $\mathcal{D}'_j = \mathcal{D}_j$, with $P[Q/x]$ instead of $(\lambda x.P)Q$, or

b) $\mathcal{D}_j$ reduces to $\mathcal{D}'_j$ at position p with redex $(\lambda x.P)Q$.

Then $\mathcal{D}$ reduces to $\langle \mathcal{D}'_1, \ldots, \mathcal{D}'_n, \cap I\rangle$ at position p with redex R.

iii) We write $\mathcal{D} \rightarrow_{\mathcal{D}} \mathcal{D}'$ if there exists a position p and redex R such that $\mathcal{D}$ reduces to $\mathcal{D}'$ at position p with redex R. If $\mathcal{D}_1 \rightarrow_{\mathcal{D}} \mathcal{D}_2 \rightarrow_{\mathcal{D}} \mathcal{D}_3$, then $\mathcal{D}_1 \rightarrow_{\mathcal{D}} \mathcal{D}_3$.

We abbreviate '$\mathcal{D}$ is strongly normalisable with respect to $\rightarrow_{\mathcal{D}}$' by '$SN(\mathcal{D})$', and use $\mathcal{SN}$ for the set of strongly normalisable derivations: $\mathcal{SN} = \{\mathcal{D} \mid SN(\mathcal{D})\}$.

The following lemma formulates the relation between derivation reduction and β-reduction.

Lemma 2.4 *Let $\mathcal{D} :: B \vdash M : \sigma$, and $\mathcal{D} \rightarrow_{\mathcal{D}} \mathcal{D}' :: B \vdash N : \sigma$, then $M \twoheadrightarrow_\beta N$.*

Proof: By the above definition. ■

The following states some standard properties of strong normalisation.

Lemma 2.5 *i)* $SN(\langle \mathcal{D}_1, \mathcal{D}_2, \rightarrow E\rangle) \Rightarrow SN(\mathcal{D}_1)$ & $SN(\mathcal{D}_2)$.

ii) $SN(\mathcal{D}_1 :: B \vdash xM_1 \cdots M_n : \sigma \rightarrow \tau)$ & $SN(\mathcal{D}_2 :: B \vdash N : \sigma) \Rightarrow SN(\langle \mathcal{D}_1, \mathcal{D}_2, \rightarrow E\rangle)$.

iii) *Let $\mathcal{D} :: B \vdash M : \sigma$ be $\langle \mathcal{D}_1 \cap \cdots \cap \mathcal{D}_n, \cap I\rangle$, so $\sigma = \sigma_1 \cap \cdots \cap \sigma_n$. If $\mathcal{D} \rightarrow_{\mathcal{D}} \mathcal{D}' :: B \vdash M' : \sigma$ at position p, then, for every $i \in \underline{n}$ there exist $\mathcal{D}'_i$ such that*

$$\text{either } \mathcal{D}_i \rightarrow_{\mathcal{D}} \mathcal{D}'_i :: B \vdash M : \sigma_i \text{ at position } p.$$

iv) *For all $i \in \underline{n}$, $SN(\mathcal{D}_1 :: B \vdash M : \sigma_i)$ if and only if $SN(\langle \mathcal{D}_1 \cap \cdots \cap \mathcal{D}_n, \cap I\rangle)$.*

v) *If $SN(\mathcal{D}_1 :: B \vdash \mathrm{C}[M[N/x]] : \sigma)$, and $SN(\mathcal{D}_2 :: B \vdash N : \rho)$, then there exists a derivation $\mathcal{D}_3$ such that $SN(\mathcal{D}_3 :: B \vdash \mathrm{C}[(\lambda y.M)N] : \sigma)$.*

Proof: Standard, using Definition *2.3*. ■

Example 2.6 Let $\mathcal{D}_1, \mathcal{D}_2$ be the derivations from Example *3.11*, and let $\mathcal{D}'_2$ be the subderivation

$$\dfrac{\dfrac{\overline{x{:}\tau, y{:}\omega{\to}\sigma \vdash y : \omega{\to}\sigma} \quad \overline{x{:}\tau, y{:}\omega{\to}\sigma \vdash xxy : \omega}}{x{:}\tau, y{:}\omega{\to}\sigma \vdash y(xxy) : \sigma}}{x{:}\tau \vdash \lambda y.y(xxy) : (\omega{\to}\sigma){\to}\sigma}$$

that occurs in $\mathcal{D}_2$. By rule $(\to E)$ we can construct:

$$\dfrac{\begin{array}{cc}\mathcal{D}_2 & \mathcal{D}_1 \\ \vdash \lambda xy.y(xxy) : \tau{\to}(\omega{\to}\rho){\to}\rho & \vdash \Theta : \tau\end{array}}{\vdash (\lambda xy.y(xxy))\Theta : (\omega{\to}\rho){\to}\rho}\,(\to E)$$

Notice that the term $(\lambda xy.y(xxy))\Theta$ has only *one* redex, that is not typed with ω; contracting it gives $\mathcal{D}'_2[\mathcal{D}_1/x{:}\tau]$, i.e.:

$$\dfrac{\dfrac{\dfrac{}{y{:}\omega{\to}\rho \vdash y : \omega{\to}\rho}\,(Ax) \quad \dfrac{}{y{:}\omega{\to}\rho \vdash (\Theta\Theta y) : \omega}\,(\cap I)}{y{:}\omega{\to}\rho \vdash y(\Theta\Theta y) : \rho}\,(\to E)}{\vdash \lambda y.y(\Theta\Theta y) : (\omega{\to}\rho){\to}\rho}\,(\to I)$$

Notice that this last derivation is in normal form, but the term $\lambda y.y(\Theta\Theta y)$ is not.

2.2 Strong normalisation result

In this subsection, we will prove a strong normalisation result for derivation reduction..

In order to prove that each derivation in '$\vdash$' is strongly normalisable with respect to $\to_{\mathcal{D}}$, a notion of computable derivations is introduced (the technique of computability predicates [21, 16] was also used in [11, 14, 1, 2, 4, 5, 7, 6]). We will show that all computable derivations are strongly normalisable with respect to derivation reduction, and then that all derivations in '$\vdash$' contain a computable component.

Definition 2.7 The Computability Predicate $\mathit{Comp}(\mathcal{D})$ is defined inductively on types by:

$$\begin{array}{ll}
\mathit{Comp}(\mathcal{D} :: B \vdash M:\varphi) & \iff \mathit{SN}(\mathcal{D}) \\
\mathit{Comp}(\mathcal{D}_1 :: B \vdash M:\alpha{\to}\beta) & \iff \\
\qquad \mathit{Comp}(\mathcal{D}_2 :: B \vdash N:\alpha) \Rightarrow \mathit{Comp}(\langle \mathcal{D}_1, \mathcal{D}_2, {\to}E\rangle :: B \vdash MN:\beta) & \\
\mathit{Comp}(\langle \mathcal{D}_1, \ldots, \mathcal{D}_n, {\cap}I\rangle :: B \vdash M:\sigma_1{\cap}\cdots{\cap}\sigma_n) & \\
\qquad \iff \forall i \in \underline{n}\,[\mathit{Comp}(\mathcal{D}_i :: B \vdash M:\sigma_i)] &
\end{array}$$

Notice that, as a special case for the third rule, we get $\mathit{Comp}(\langle {\cap}I\rangle :: B \vdash M:\omega)$

The following lemma formulates the relation between the computability predicate and the relation $\leq$ on derivations, and is crucial for the proof of Theorem *2.11*. The main difference between the solution of [3] and the one presented here lies in the fact that here we need to prove this lemma, whereas in [3], it is not needed at all.

Lemma 2.8 If $\mathit{Comp}(\mathcal{D} :: B \vdash M:\sigma)$*, and* $\mathcal{D} \leq \mathcal{D}'$*, then* $\mathit{Comp}(\mathcal{D}')$.

Proof: By induction on the structure of types. Notice that, by Lemma *1.8*, $\mathcal{D}' = B' \vdash M:\sigma'$, with $B' \leq B$, $\sigma \leq \sigma'$.

We distinguish two cases:

$(\sigma' \in \mathcal{T}_s)$: $(\sigma = \varphi)$: Since $\varphi \leq \sigma'$, also $\sigma' = \varphi$, and the result is immediate.

$(\sigma = \alpha{\to}\beta)$: Then $\sigma' = \rho{\to}\mu$, with $\rho \leq \alpha, \beta \leq \mu$, and let $\mathcal{D}' :: B \vdash M:\rho{\to}\mu$.

To show $\mathit{Comp}(\mathcal{D}')$, by Definition *2.7*, we assume $\mathit{Comp}(\mathcal{D}_0 :: B \vdash N:\rho)$, and use this to show that $\langle \mathcal{D}', \mathcal{D}_0, {\to}E\rangle :: B \vdash MN:\mu$.

Since $\mathcal{D}_0 \leq \mathcal{D}'_0 :: B \vdash N:\alpha$, from $\mathit{Comp}(\mathcal{D}_0)$ we get $\mathit{Comp}(\mathcal{D}'_0)$ by induction. Assuming $\mathit{Comp}(\mathcal{D} :: B \vdash M:\alpha{\to}\beta)$, $\mathit{Comp}(\langle \mathcal{D}, \mathcal{D}'_0, {\to}E\rangle :: B \vdash MN:\beta)$ follows by Definition *2.7*. Then $\langle \mathcal{D}, \mathcal{D}'_0, {\to}E\rangle \leq \langle \mathcal{D}', \mathcal{D}_0, {\to}E\rangle :: B \vdash MN:\mu$, and we get $\mathit{Comp}(\langle \mathcal{D}', \mathcal{D}_0, {\to}E\rangle)$, again by induction. So $\mathit{Comp}(\mathcal{D} :: B \vdash M:\rho{\to}\mu)$, by Definition *2.7*.

$(\sigma = \sigma_1{\cap}\cdots{\cap}\sigma_n)$: If $\mathit{Comp}(\mathcal{D} :: B \vdash M:\sigma_1{\cap}\cdots{\cap}\sigma_n)$, then $\mathcal{D} = \langle \mathcal{D}_1, \ldots, \mathcal{D}_n, {\cap}I\rangle$, by Definition *2.7*, and $\mathit{Comp}(\mathcal{D}_i :: B \vdash M:\sigma_i)$ for $i \in \underline{n}$. Since $\sigma_1{\cap}\cdots{\cap}\sigma_n \leq \sigma'$, by Property *1.3*, there exists $i_j \in \underline{n}$ such that $\sigma_{i_j} \leq \sigma'$. Then $\mathcal{D} \leq \mathcal{D}_{i_j} :: B \vdash M:\tau_j$ and, by induction, $\mathit{Comp}(\mathcal{D}_{i_j})$.

$(\sigma' = \sigma_1{\cap}\cdots{\cap}\sigma_n)$: If $\mathit{Comp}(\mathcal{D} :: B \vdash M:\sigma_1{\cap}\cdots{\cap}\sigma_n)$, then, by Definition *2.7*, for every $i \in \underline{n}$ there exist $\mathcal{D}_i$ such that $\mathit{Comp}(\mathcal{D}_i :: B \vdash M:\sigma_i)$, and $\mathcal{D} = \langle \mathcal{D}_1, \ldots, \mathcal{D}_n, {\cap}I\rangle$. Since $\sigma_1{\cap}\cdots{\cap}\sigma_n \leq \tau$, by Property *1.3*, $\tau = \tau_1{\cap}\cdots{\cap}\tau_m$, and for all $j \in \underline{m}$ there exists $i_j \in \underline{n}$ such that $\sigma_{i_j} \leq \tau_j$. Since $\mathcal{D}_i \leq \mathcal{D}_{i_j} :: B \vdash M:\tau_j$, by induction, $\mathit{Comp}(\mathcal{D}_{i_j})$, and, by Definition *2.7*,

$$\mathit{Comp}(\langle \mathcal{D}_{i_1}, \ldots, \mathcal{D}_{i_m}, {\cap}I\rangle :: B \vdash M:\tau_1{\cap}\cdots{\cap}\tau_m)$$

■

We will now prove that *Comp* satisfies the standard properties of computability predicates, being that computability implies strong normalisation, and that, for the so-called *neutral* objects, also the converse holds.

Lemma 2.9 *i)* $\mathit{Comp}(\mathcal{D} :: B \vdash M : \sigma) \Rightarrow \mathit{SN}(\mathcal{D})$.
ii) $\mathit{SN}(\mathcal{D} :: B \vdash xM_1 \cdots M_m : \sigma) \Rightarrow \mathit{Comp}(\mathcal{D})$.

Proof: By simultaneous induction on the structure of types.

$(\sigma = \varphi)$: Directly by Definition *2.7*.

$(\sigma = \alpha \to \beta)$: *i*) Let x be a variable not appearing in M, and let $\mathcal{D}' :: x{:}\alpha \vdash x : \alpha$, then, by induction *(ii)*, $\mathit{Comp}(\mathcal{D}')$. Assume, without loss of generality, that $x{:}\alpha \in B$. By assumption, $\mathit{Comp}(\mathcal{D} :: B \vdash M : \alpha \to \beta)$, and $\mathit{Comp}(\langle \mathcal{D}, \mathcal{D}', \to E \rangle :: B \vdash Mx : \beta)$ by Definition *2.7*. Then, by induction *(i)*, $\mathit{SN}(\langle \mathcal{D}, \mathcal{D}', \to E \rangle)$, so also $\mathit{SN}(\mathcal{D})$.

ii) Assume $\mathit{Comp}(\mathcal{D}' :: B \vdash N : \alpha)$, then by induction *(i)*, $\mathit{SN}(\mathcal{D}')$. Then also by Lemma *2.5 (ii)*, $\mathit{SN}(\langle \mathcal{D}, \mathcal{D}', \to E \rangle :: B \vdash xM_1 \cdots M_m N : \beta)$. Then, by induction *(ii)*, $\mathit{Comp}(\langle \mathcal{D}, \mathcal{D}', \to E \rangle)$, so by Definition *2.7*, $\mathit{Comp}(\mathcal{D})$.

$(\sigma = \sigma_1 \cap \cdots \cap \sigma_n)$: Easy, using Definition *2.7*, Lemma *2.5 (iv)*, and induction. ■

The following theorem (*2.11*) shows that, if the instances of rule (Ax) are to be replaced by computable derivations, then the result itself will be computable. Before coming to this result, first two auxiliary lemmas are proved.

The first lemma shows that the predicate is closed for subject-expansion.

Lemma 2.10 *If* $\mathit{Comp}(\mathcal{D}' :: B \vdash Q : \nu)$ *and* $\mathit{Comp}(\mathcal{D}[\mathcal{D}'/y{:}\nu] :: B \vdash M[Q/y]\overrightarrow{P} : \sigma)$, *then there exists a derivation* $\mathcal{D}''$ *such that* $\mathit{Comp}(\mathcal{D}'' :: B \vdash (\lambda y.M)Q\overrightarrow{P} : \sigma)$.

Proof: By induction on the structure of types.

i) $\sigma = \varphi$.
$\mathit{Comp}(\mathcal{D}[\mathcal{D}'/y{:}\nu] :: B \vdash M[Q/y]\overrightarrow{P} : \varphi)$ & $\mathit{Comp}(\mathcal{D}' :: B \vdash Q : \nu) \Rightarrow$ (*2.9 (i)*)
$\mathit{SN}(\mathcal{D}[\mathcal{D}'/y{:}\nu])$ & $\mathit{SN}(\mathcal{D}') \Rightarrow$ (*2.5 (v)*)
$\exists \mathcal{D}''\ [\mathit{SN}(\mathcal{D}'' :: B \vdash (\lambda y.M)Q\overrightarrow{P} : \varphi)] \Rightarrow$ (*2.7*)
$\exists \mathcal{D}''\ [\mathit{Comp}(\mathcal{D}'' :: B \vdash (\lambda y.M)Q\overrightarrow{P} : \varphi)]$.

ii) $\sigma = \alpha \to \beta$.
$\mathit{Comp}(\mathcal{D}_1 :: B \vdash N : \alpha)$ & $\mathit{Comp}(\mathcal{D}_2 :: B \vdash Q : \nu) \Rightarrow$ (*2.7*)
$\mathit{Comp}(\langle \mathcal{D}[\mathcal{D}'/y{:}\nu], \mathcal{D}_2, \to E \rangle :: B \vdash M[Q/y]\overrightarrow{P}N : \beta) \Rightarrow$ (*IH*)
$\exists \mathcal{D}''[\mathit{Comp}(\langle \mathcal{D}'', \mathcal{D}_2, \to E \rangle :: B \vdash (\lambda y.M)Q\overrightarrow{P}N : \beta) \Rightarrow$ (*2.7*)
$\exists \mathcal{D}''[\mathit{Comp}(\mathcal{D}'' :: B \vdash (\lambda y.M)Q\overrightarrow{P} : \alpha \to \beta)]$

iii) $\sigma = \sigma_1 \cap \cdots \cap \sigma_n$. By induction and Definition *2.7*.

We now come to the Replacement Theorem, i.e. the proof that for every derivation, if the instances of rule (Ax) in the derivation are to be replaced by computable derivations, then the result itself will be computable.

We will use an abbreviated notation, and write $[\overrightarrow{N_i/x_i}]$ for $[N_1/x_1,\ldots,N_n/x_n]$, etc.

Theorem 2.11 *Let $B = x_1{:}\mu_1,\ldots,x_n{:}\mu_n$, $\mathcal{D} :: B \vdash M{:}\sigma$, and, for every $i \in \underline{n}$, there are $\mathcal{D}^i, N^i$ such that Comp$(\mathcal{D}^i :: B' \vdash N^i{:}\mu_i)$. Then*

$$\mathit{Comp}(\mathcal{D}[\overrightarrow{\mathcal{D}_i/x_i{:}\mu_i}] :: B' \vdash M[\overrightarrow{N_i/x_i}]{:}\sigma).$$

Proof: By induction on the structure of derivations.

(Ax) : Then $M \equiv x_i$, for some $i \in \underline{n}$, with $\mu_i \leq \sigma$. Since $\mathcal{D}^i \leq \mathcal{D}' :: B' \vdash N^i{:}\sigma$, from *Comp*$(\mathcal{D}^i)$, by Lemma *2.8*, *Comp*$(\mathcal{D}')$. Notice that $\mathcal{D}' = (\langle Ax\rangle :: B \vdash x{:}\sigma)[\overrightarrow{\mathcal{D}_i/x_i{:}\mu_i}]$

($\cap I$) : $\sigma = \sigma_1 \cap \cdots \cap \sigma_m$, and, for $j \in \underline{m}$, there exists $\mathcal{D}_j$, such that $\mathcal{D}_j :: B \vdash M{:}\sigma_j$ and $\mathcal{D} = \langle \mathcal{D}_1,\ldots,\mathcal{D}_m, \cap I\rangle$. Let, for $j \in \underline{m}$,

$$\mathcal{D}'_j = \mathcal{D}_j[\overrightarrow{\mathcal{D}_i/x_i{:}\mu_i}] :: B \vdash M[\overrightarrow{N_i/x_i}]{:}\sigma_j,$$

then, by induction, *Comp*$(\mathcal{D}'_j)$. Let $\mathcal{D}' = \langle \mathcal{D}'_1,\ldots,\mathcal{D}'_m, \cap I\rangle$, then, by Definition *2.7*,

$$\mathit{Comp}(\mathcal{D}' :: B \vdash M[\overrightarrow{N_i/x_i}]{:}\sigma_1 \cap \cdots \cap \sigma_m),$$

and $\mathcal{D}' = \mathcal{D}[\overrightarrow{\mathcal{D}_i/x_i{:}\mu_i}]$.

($\rightarrow I$) : Then $\sigma = \rho \rightarrow \tau$, and $\mathcal{D} = \langle \mathcal{D}_1 :: B, y{:}\rho \vdash M'{:}\tau, \rightarrow I\rangle :: B \vdash \lambda y.M'{:}\rho \rightarrow \tau$.

$$\begin{array}{lll}
\forall j \in \underline{m}\ [\mathcal{D}_j :: B_j \vdash M_j{:}\sigma_j]\ \&\ \mathit{Comp}(\mathcal{D}_2 :: B' \vdash P{:}\rho) & \Rightarrow & (\mathit{IH})\\
\mathit{Comp}(\mathcal{D}_1[\overrightarrow{\mathcal{D}_i/x_i{:}\mu_i}, \mathcal{D}_2/y{:}\rho] :: B' \vdash M[\overrightarrow{N/x}, P/y]{:}\tau) & \Rightarrow & (2.10)\\
\mathit{Comp}(\langle\langle \mathcal{D}_1[\overrightarrow{\mathcal{D}_i/x_i{:}\mu_i}], \rightarrow I\rangle, \mathcal{D}_2, \rightarrow E\rangle :: B' \vdash (\lambda y.M[\overrightarrow{N_i/x_i}])P{:}\tau) & \Rightarrow & (2.7)\\
\mathit{Comp}(\langle \mathcal{D}_1[\overrightarrow{\mathcal{D}_i/x_i{:}\mu_i}], \rightarrow I\rangle :: B' \vdash \lambda y.M[\overrightarrow{N_i/x_i}]{:}\rho \rightarrow \tau). & &
\end{array}$$

and $\mathcal{D}' = \langle \mathcal{D}_1[\overrightarrow{\mathcal{D}_i/x_i{:}\mu_i}], \rightarrow I\rangle = \mathcal{D}[\overrightarrow{\mathcal{D}_i/x_i{:}\mu_i}]$.

($\rightarrow E$) : Then $M \equiv M_1M_2$, and there are $\mathcal{D}_1, \mathcal{D}_2$, and τ such that $\mathcal{D} = \langle \mathcal{D}_1, \mathcal{D}_2, \rightarrow E\rangle$, $\mathcal{D}_1 :: B \vdash M_1{:}\tau \rightarrow \sigma$, and $\mathcal{D}_2 :: B \vdash M_2{:}\tau$. Let

$$\begin{array}{l}
\mathcal{D}'_1 = \mathcal{D}_1[\overrightarrow{\mathcal{D}_i/x_i{:}\mu_i}] :: B' \vdash M_1[\overrightarrow{N_i/x_i}]{:}\tau \rightarrow \sigma, \text{ and}\\
\mathcal{D}'_2 = \mathcal{D}_2[\overrightarrow{\mathcal{D}_i/x_i{:}\mu_i}] :: B' \vdash M_2[\overrightarrow{N_i/x_i}]{:}\tau,
\end{array}$$

then, by induction, *Comp*$(\mathcal{D}'_1)$, and *Comp*$(\mathcal{D}'_2)$, and by Definition *2.7*,

$$\mathit{Comp}(\langle \mathcal{D}'_1, \mathcal{D}'_2, \rightarrow E\rangle :: B' \vdash (M_1M_2)[\overrightarrow{N_i/x_i}]{:}\sigma),$$

Notice that $\langle \mathcal{D}_1, \mathcal{D}_2, \rightarrow E\rangle[\overrightarrow{\mathcal{D}_i/x_i{:}\mu_i}] = \langle \mathcal{D}'_1, \mathcal{D}'_2, \rightarrow E\rangle$. ■

Using this last result, we are now able to prove a strong normalisation result for derivation reduction in '$\vdash$'.

Theorem 2.12 *If $\mathcal{D} :: B \vdash M : \sigma$, then SN$(\mathcal{D})$.*

Proof: By Lemma *2.9 (ii)*, for every $x_i{:}\tau_i \in B$, there exists $\mathcal{D}_{x_i} = \langle Ax \rangle :: x_i{:}\tau_i \vdash x_i : \tau_i$ such that *Comp*$(\mathcal{D}_{x_i})$, so by Theorem *2.11*, *Comp*$(\mathcal{D}[\overrightarrow{\mathcal{D}_{x_i}/x_i{:}\tau_i}] :: B \vdash M[\overrightarrow{x_i/x_i}] : \sigma)$. Notice that $M[\overrightarrow{x_i/x_i}] = M$ and $\mathcal{D}[\overrightarrow{\mathcal{D}_{x_i}/x_i{:}\tau_i}] = \mathcal{D}$, and by Theorem *2.11*, *SN*$(\mathcal{D})$. ■

3 Approximation and head-normalisation

In this section, we will conclude the main contribution of this paper by showing two main results, that are both direct consequences of the strong normalisation result proved in Section *2.2*. Both results have been proven in the past, at least partially, in [1, 2]. In fact, some of the theorems and lemmas presented here were already presented in those papers and are repeated here, for completeness, with their proofs.

3.1 Approximate normal forms

We will now show that the above strong normalisation result leads to the approximation theorem, for which we will prepare the ground by introducing the necessary concepts.

The notion of approximant for lambda terms was first presented in [22], and is defined using the notion of terms in $\lambda\bot$-normal form (like in [8], $\bot$ (called *bottom*) is used, instead of ω; also, the symbol $\sqsubseteq$ is used as a relation on $\lambda\bot$-terms, inspired by a similar relation defined on Böhm-trees in [8]).

Definition 3.1 *i*) The set of $\lambda\bot$ *-terms* is defined as the set λ of lambda terms, by:

$$M \ ::= \ x \mid \bot \mid \lambda x.M \mid M_1 M_2$$

ii) The notion of reduction $\rightarrow_{\beta\bot}$ is defined as $\rightarrow_\beta$, extended by:

$$\begin{array}{rcl} \lambda x.\bot & \rightarrow_{\beta\bot} & \bot \\ \bot M & \rightarrow_{\beta\bot} & \bot. \end{array}$$

iii) The set of *normal forms for elements of $\lambda\bot$ with respect to $\rightarrow_{\beta\bot}$*, is the set $\mathcal{N}$ of $\lambda\bot$-*normal forms* or *approximate normal forms*, ranged over by A, inductively defined by:

$$A \ ::= \ \bot \mid \lambda x.A \ (A \neq \bot) \mid xA_1 \cdots A_n \ (n \geq 0)$$

The rules of the system '$\vdash$' are generalized to terms containing $\bot$ by allowing for the terms to be elements of $\lambda\bot$. Notice that, because type assignment is almost syntax directed, if $\bot$ occurs in a term M and $\mathcal{D} :: B \vdash M:\sigma$, then in $\mathcal{D}$, $\bot$ appears in a position where the rule $(\cap I)$ is used with $n = 0$. Moreover, the terms $\lambda x.\bot$ and $\bot M_1 \cdots M_n$ are typeable by ω only.

Definition 3.2 *i*) The relation $\sqsubseteq \subseteq \lambda\bot^2$ is defined by:

$$\begin{array}{rcl} \bot & \sqsubseteq & M \\ x & \sqsubseteq & x \\ M \sqsubseteq M' & \Rightarrow & \lambda x.M \sqsubseteq \lambda x.M' \\ M_1 \sqsubseteq M_1' \;\&\; M_2 \sqsubseteq M_2' & \Rightarrow & M_1M_2 \sqsubseteq M_1'M_2'. \end{array}$$

If $A \in \mathcal{N}$, $M \in \lambda$, and $A \sqsubseteq M$, then A is called a *direct approximant* of M.

ii) The relation $\underset{\sim}{\sqsubseteq} \subseteq \mathcal{N} \times \lambda$ is defined by:

$$A \underset{\sim}{\sqsubseteq} M \iff \exists M' =_\beta M[A \underset{\sim}{\sqsubseteq} M'].$$

iii) If $A \underset{\sim}{\sqsubseteq} M$, then A is called an *approximant* of M, and $\mathcal{A}(M) = \{ A \in \mathcal{N} \mid A \underset{\sim}{\sqsubseteq} M \}$.

Lemma 3.3 $B \vdash M:\sigma \;\&\; M \sqsubseteq M' \;\Rightarrow\; B \vdash M':\sigma$.

Proof: By easy induction on the definition of $\sqsubseteq$. ■

The following definition introduces an operation of join on $\lambda\bot$-terms.

Definition 3.4 *i*) On $\lambda\bot$, the partial mapping $\sqcup : \lambda\bot \times \lambda\bot \to \lambda\bot$ is defined by:

$$\begin{array}{rcl} \bot \sqcup M & \equiv & M \sqcup \bot \equiv M \\ x \sqcup x & \equiv & x \\ (\lambda x.M) \sqcup (\lambda x.N) & \equiv & \lambda x.(M \sqcup N) \\ (M_1M_2) \sqcup (N_1N_2) & \equiv & (M_1 \sqcup N_1)\,(M_2 \sqcup N_2) \end{array}$$

$\sqcup$ is pronounced *join*.

ii) If $M \sqcup N$ is defined, then M and N are called *compatible*.

From now on, to shorten proofs, $\bot$ will be the same as the empty join, i.e. if $M \equiv M_0 \sqcup \cdots \sqcup M_n$, and $n = 0$, then $M \equiv \bot$.

The last alternative in the definition of $\sqcup$ defines the join on applications in a more general way than Scott's, that would state that $(M_1M_2) \sqcup (N_1N_2) \sqsubseteq (M_1 \sqcup N_1)(M_2 \sqcup N_2)$, since it

is not always sure if a join of two arbitrary terms exists. However, we will use our more general definition only on terms that are compatible, so the conflict is only apparent.

The following lemma shows that $\sqcup$ acts as least upper bound of compatible terms.

Lemma 3.5 *i) If $M_1 \sqsubseteq M$, and $M_2 \sqsubseteq M$, then $M_1 \sqcup M_2$ is defined, and:*
$$M_1 \sqsubseteq M_1 \sqcup M_2, M_2 \sqsubseteq M_1 \sqcup M_2, \text{ and } M_1 \sqcup M_2 \sqsubseteq M.$$
ii) If $M \sqsubseteq M_i$, for $i \in \underline{n}$, then $M \sqsubseteq M_1 \sqcup \cdots \sqcup M_n$.
iii) If $M \sqsubseteq N$, and $N \sqsubseteq P$, then $M \sqsubseteq P$.
iv) If $M \sqsubseteq M_1 M_2$, then there are M_3, M_4 such that $M = M_3 M_4$, and $M_3 \sqsubseteq M_1, M_4 \sqsubseteq M_2$.

Proof: By induction on the definition of $\sqsubseteq$.

i) If $M_1 \equiv \bot$, then $M_1 \sqcup M_2 \equiv M_2$, so $M_1 \sqsubseteq M_1 \sqcup M_2, M_2 \sqsubseteq M_1 \sqcup M_2$, and $M_1 \sqcup M_2 \sqsubseteq M_2 \sqsubseteq M$. (The case $M_2 \equiv \bot$ goes similarly.)

ii) If $M_1 \equiv x$, and $M_2 \equiv x$, then $M_1 \sqcup M_2 \equiv x$. Obviously, $x \sqsubseteq x \sqcup x$, $x \sqsubseteq x \sqcup x$, and $x \sqcup x \sqsubseteq x$.

iii) If $M_1 \equiv \lambda x.N_1$, and $M_2 \equiv \lambda x.N_2$, then $M \equiv \lambda x.N$, $N_1 \sqsubseteq N$, $N_2 \sqsubseteq N$. Then, by induction, $N_1 \sqsubseteq N_1 \sqcup N_2$, $N_2 \sqsubseteq N_1 \sqcup N_2$, and $N_1 \sqcup N_2 \sqsubseteq N$. Also $\lambda x.N_1 \sqsubseteq \lambda x.N_1 \sqcup N_2$, $\lambda x.N_2 \sqsubseteq \lambda x.N_1 \sqcup N_2$, and $\lambda x.N_1 \sqcup N_2 \sqsubseteq \lambda x.N$. To conclude, notice that $\lambda x.N_1 \sqcup N_2 \equiv (\lambda x.N_1) \sqcup (\lambda x.N_2)$.

iv) If $M_1 \equiv P_1 Q_1$, and $M_2 \equiv P_2 Q_2$, then $M \equiv PQ$, $P_1 \sqsubseteq P$, $Q_1 \sqsubseteq Q$, $P_2 \sqsubseteq P$, $Q_2 \sqsubseteq Q$. By induction, we know $P_1 \sqsubseteq P_1 \sqcup P_2$, $P_2 \sqsubseteq P_1 \sqcup P_2$, and $P_1 \sqcup P_2 \sqsubseteq P$, as well as $Q_1 \sqsubseteq Q_1 \sqcup Q_2$, $Q_2 \sqsubseteq Q_1 \sqcup Q_2$, and $Q_1 \sqcup Q_2 \sqsubseteq Q$. Then also $P_1 Q_1 \sqsubseteq (P_1 \sqcup P_2)(Q_1 \sqcup Q_2)$, $P_2 Q_2 \sqsubseteq (P_1 \sqcup P_2)(Q_1 \sqcup Q_2)$, and $(P_1 \sqcup P_2)(Q_1 \sqcup Q_2) \sqsubseteq PQ$. To conclude, notice that $(P_1 \sqcup P_2)(Q_1 \sqcup Q_2) \equiv (P_1 Q_1) \sqcup (P_2 Q_2)$. ■

3.2 The $\bot$ type assignment system

A first approach to the problem dealt with in this paper was to try to show that, when no redex is typed with ω, then reduction on typeable terms is strongly normalisable, so in particular, terms typeable in '$\vdash_\bot$' then should be strongly normalisable. In detail, this implied that, in this system, the use of ω is restricted to $\bot$ *only*, instead of allowing any term to be typeable by ω. However, perhaps surprisingly, the strong normalisation result turned out not to hold.

We will start this section by defining this $\bot$-system in detail, also because it is of use in the last section of this paper.

Definition 3.6 $\perp$*-type assignment* and $\perp$*-derivations* are defined by the following natural deduction system (where all types displayed are strict, except σ in the rules (Ax), $(\to I)$, and $(\to E)$):

$$(Ax): \frac{}{B, x{:}\sigma \vdash_\perp x:\tau}\,(\sigma \leq \tau) \qquad (\cap I): \frac{B \vdash_\perp M_1:\sigma_1 \quad \ldots \quad B \vdash_\perp M_n:\sigma_n}{B \vdash_\perp M_1 \sqcup \cdots \sqcup M_n : \sigma_1 \cap \cdots \cap \sigma_n}\,(n \geq 0)$$

$$(\to I): \frac{B, x{:}\sigma \vdash_\perp M:\tau}{B \vdash_\perp \lambda x.M : \sigma \to \tau} \qquad (\to E): \frac{B \vdash_\perp M:\sigma \to \tau \quad B \vdash_\perp N:\sigma}{B \vdash_\perp MN:\tau}$$

We write $B \vdash_\perp M:\sigma$ if this statement is derivable using a $\perp$-derivation.

Notice that, by rule $(\cap I)$, $B \vdash_\perp \perp:\omega$, and that this is the only way to assign ω to a term. In particular, since by the remark made after Definition *3.1*, the terms $\lambda x.\perp$ and $\perp M_1 \cdots M_n$ are typeable in '$\vdash$' by ω only, these terms are *not* typeable in '$\vdash_\perp$'. So a term typeable in '$\vdash_\perp$' contains no redexes of those forms.

An important point to note is that the operation of join is used for rule $(\cap I)$ in the above definition. Of course it is possible to define a notion of type assignment on terms – that allows for the use of ω for $\perp$ only – without the join operation, but the system obtained in that way would not be expressive enough; the idea is to, in derivations of the full system, replace subterms typed by ω by $\perp$. In the presence of intersection types, this has to be done with care.

The relation between the two notions of type assignment '$\vdash$' and '$\vdash_\perp$' is formulated by:

Lemma 3.7 *If* $\mathcal{D} :: B \vdash_\perp M:\sigma$, *then* $\mathcal{D} :: B \vdash M:\sigma$.

Proof: By induction on the structure of derivations in '$\vdash_\perp$'.

(Ax) : Immediate.

$(\cap I)$: Then $M = M_1 \sqcup \cdots \sqcup M_n$ for some $M_1, \ldots, M_n$, and, for every $i \in \underline{n}$, $B \vdash_\perp M_i:\sigma_i$. Then, by induction, for every $i \in \underline{n}$, $B \vdash M_i:\sigma_i$, so by *3.5 (i)* & *3.3*, for every $i \in \underline{n}$, $B \vdash M:\sigma_i$, so by $(\cap I)$, $B \vdash M:\sigma_1 \cap \cdots \cap \sigma_n$.

$(\to I)$: Then $M \equiv \lambda x.M'$, and $\sigma = \alpha \to \beta$, and $B, x{:}\alpha \vdash_\perp M':\beta$. So $B, x{:}\alpha \vdash M':\beta$ by induction, so $B \vdash \lambda x.M':\alpha \to \beta$ by $(\to I)$.

$(\to E)$: Then $M \equiv M_1 M_2$, and there exists τ such that $B \vdash_\perp M_1:\tau \to \sigma$, and $B \vdash_\perp M_2:\tau$. Then, by induction, $B \vdash M_1:\tau \to \sigma$, and $B \vdash M_2:\tau$, so by $(\to E)$ we get $B \vdash M_1 M_2:\sigma$. ■

Lemma 3.8 *If* $\mathcal{D} :: B \vdash M:\sigma$, *then there are* $M' \sqsubseteq M$, *and* $\mathcal{D} :: B \vdash_\perp M':\sigma$.

Proof: By induction on the structure of derivations in '$\vdash$'.

(Ax) : Immediate.

$(\cap I)$: Then $\sigma = \sigma_1 \cap \cdots \cap \sigma_n$ and, for every $i \in \underline{n}$, $B \vdash M : \sigma_i$, and, by induction, for every $i \in \underline{n}$ there are $M_i \sqsubseteq M$ such that $B \vdash_\perp M_i : \sigma_i$. Then $B \vdash_\perp M_1 \sqcup \cdots \sqcup M_n : \sigma_1 \cap \cdots \cap \sigma_n$, by $(\cap I)$. By Lemma *3.5 (i)*, $M_1 \sqcup \cdots \sqcup M_n \sqsubseteq M$.

$(\rightarrow I)$: Then $M \equiv \lambda x.M_1$, and $\sigma = \alpha \rightarrow \beta$, and $B, x{:}\alpha \vdash M_1 : \beta$. So, by induction, there exists $M_1' \sqsubseteq M_1$ such that $B, \alpha \vdash M' : \beta$. By rule $(\rightarrow I)$ we obtain $B' \vdash \lambda x.M_1' : \alpha \rightarrow \beta$. Notice that $\lambda x.M_1' \sqsubseteq \lambda x.M_1$.

$(\rightarrow E)$: Then $M \equiv M_1 M_2$, and there is a τ such that $B \vdash M_1 : \tau \rightarrow \sigma$, and $B \vdash M_2 : \tau$. Then, by induction, there are $M_1' \sqsubseteq M_1$ and $M_2' \sqsubseteq M_2$, such that $B \vdash_\perp M_1' : \tau \rightarrow \sigma$, $B \vdash_\perp M_2' : \tau$. Then by $(\rightarrow E)$, $B \vdash_\perp M_1' M_2' : \sigma$. Notice that $M_1' M_2' \sqsubseteq M_1 M_2$. ■

Notice that the derivation for $B \vdash_\perp M' : \sigma$ is not truly equal to $\mathcal{D}$ in that the term involved is different; however, they are equal in structure in the sense of applied rules. Morever, notice that the case $\sigma = \omega$ from part *3.8* is hidden in the case $(\cap I)$ of the proof. Then $n = 0$, and $M_1 \sqcup \cdots \sqcup M_n = \perp$.

Using these relations, the following property becomes easy.

Proposition 3.9 (SUBJECT REDUCTION) *If $B \vdash_\perp M : \tau$ and $M \twoheadrightarrow_\beta N$, then there exists $N' \sqsubseteq N$ such that $B \vdash_\perp N' : \tau$.*

Proof: If $B \vdash_\perp M : \tau$, by Lemma *3.7*, also $B \vdash M : \tau$. Since $M \twoheadrightarrow_\beta N$, by Theorem *1.5 (iii)*, also $B \vdash N : \tau$. Then by Lemma *3.8*, there exists a $N' \sqsubseteq N$ such that $B \vdash_\perp N' : \tau$. ■

To prepare the characterisation of terms by their assignable types, first we prove that a term in $\lambda\perp$-normal form is typeable without ω, if and only if it does not contain $\perp$. This forms the basis for the result that all normalisable terms are typeable without ω.

Lemma 3.10 ([2]) *If $B \vdash A : \sigma$, and B, σ are ω-free, then A is $\perp$-free.*

Proof: By induction on A. As before, only the part $\sigma \in \mathcal{T}_s$ is shown.

i) $A \equiv x$. Immediate.

ii) $A \equiv \perp$. Impossible, by the observation made after Definition *3.6*.

iii) $A \equiv \lambda x.A'$. By $(\rightarrow I)$ there are α, β such that $\sigma = \alpha \rightarrow \beta$, and $B, x{:}\alpha \vdash A' : \beta$. Of course also $B, x{:}\alpha$, and β are ω-free, so by induction, A' is $\perp$-free, so also $\lambda x.A'$ is $\perp$-free.

iv) $A \equiv x A_1 \cdots A_n$. Then by $(\rightarrow E)$ and $(\leq)$ there are σ_i $(i \in \underline{n})$, $\tau_1, \ldots, \tau_n$, τ, such that for every $i \in \underline{n}$, $B \vdash A_i : \sigma_i$, and $x{:}\tau_1 \rightarrow \cdots \rightarrow \tau_n \rightarrow \tau \in B$, and $\tau_1 \rightarrow \cdots \rightarrow \tau_n \rightarrow \tau \leq \sigma_1 \rightarrow \cdots \rightarrow \sigma_n \rightarrow \sigma$. So, especially, for every $i \in \underline{n}$, $\sigma_i \leq \tau_i$. By Property *1.5 (i)*, also for

every $i \in \underline{n}$, $B \vdash A_i : \tau_i$. Since each τ_i occurs in B, all are ω-free, so by induction each A_i is $\bot$-free. Then also $xA_1 \cdots A_n$ is $\bot$-free. ■

As already shown in [1], when restricting the system '$\vdash$' to one that does not use the type constant ω at all, all typeable terms are strongly normalisable. This could give rise to the idea that possible non-normalising subterms can only occur in derivations in '$\vdash$' in subderivations that derive $B \vdash M : \omega$. This is not the case: although in the system '$\vdash_\bot$' no redex can be typed with ω, typeable terms need not be strongly normalizing, as clearly illustrated by the following example.

Example 3.11 (CF. [3]) Take $\Theta = \lambda xy.y(xxy)$, then $\Theta\Theta$ is typeable in '$\vdash_\bot$'. First we derive $\mathcal{D}_1$ (where $B = \{x{:}(\alpha{\to}\beta{\to}\gamma){\cap}\alpha, y{:}(\gamma{\to}\delta){\cap}\beta\}$):

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{B \vdash_\bot x : \alpha{\to}\beta{\to}\gamma} \quad \dfrac{}{B \vdash_\bot x : \alpha}}{B \vdash_\bot xx : \beta{\to}\gamma} \quad \dfrac{}{B \vdash_\bot y : \beta}}{B \vdash_\bot xxy : \gamma}}{B \vdash_\bot y : \gamma{\to}\delta} \quad \Big/}{B \vdash_\bot y(xxy) : \delta}}{B\backslash y \vdash_\bot \lambda y.y(xxy) : ((\gamma{\to}\delta){\cap}\beta){\to}\delta}}{\vdash_\bot \Theta : ((\alpha{\to}\beta{\to}\gamma){\cap}\alpha){\to}((\gamma{\to}\delta){\cap}\beta){\to}\delta}$$

Let $\tau = ((\alpha{\to}\beta{\to}\gamma){\cap}\alpha){\to}((\gamma{\to}\delta){\cap}\beta){\to}\delta$ (i.e. the type derived in the previous derivation), then we can derive $\mathcal{D}_2$:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x{:}\tau, y{:}\omega{\to}\sigma \vdash_\bot y : \omega{\to}\sigma} \quad \dfrac{}{x{:}\tau, y{:}\omega{\to}\sigma \vdash_\bot \bot : \omega}}{x{:}\tau, y{:}\omega{\to}\sigma \vdash_\bot y\bot : \sigma}}{x{:}\tau \vdash_\bot \lambda y.y\bot : (\omega{\to}\sigma){\to}\sigma}}{\vdash_\bot \lambda xy.y\bot : \tau{\to}(\omega{\to}\sigma){\to}\sigma}$$

From these, by $(\cap I)$, since $\lambda xy.y\bot \sqsubseteq \Theta$, we obtain a derivation for $\vdash_\bot \Theta : (\tau{\to}(\omega{\to}\sigma){\to}\sigma) \cap \tau$. Let $B = x{:}(\tau{\to}(\omega{\to}\sigma){\to}\sigma) \cap \tau, y{:}\omega{\to}\sigma$, then also:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{B \vdash_\perp x:\tau\to(\omega\to\sigma)\to\sigma}\quad\dfrac{}{B \vdash_\perp x:\tau}}{B \vdash_\perp xx:(\omega\to\sigma)\to\sigma}\quad\dfrac{}{B \vdash_\perp y:\omega\to\sigma}}{B \vdash_\perp xxy:\sigma}}{\dfrac{B \vdash_\perp y:\sigma\to\sigma}{B \vdash_\perp y(xxy):\sigma}}}{\dfrac{x:(\tau\to(\omega\to\sigma)\to\sigma)\cap\tau \vdash_\perp \lambda y.y(xxy):(\omega\to\sigma)\to\sigma}{\vdash_\perp \Theta:((\tau\to(\omega\to\sigma)\to\sigma)\cap\tau)\to(\omega\to\sigma)\to\sigma}}$$

Then, by $(\to E)$, we obtain $\vdash_\perp \Theta\Theta:(\omega\to\sigma)\to\sigma$. Notice that this term is not strongly normalisable, since

$$\Theta\Theta \to_\beta \lambda y.(\Theta\Theta y) \twoheadrightarrow_\beta \lambda y.(y(\Theta\Theta y)) \twoheadrightarrow_\beta \cdots.$$

and that, in particular, the redex is not typed with ω. Moreover, all its reducts are typeable in '$\vdash_\perp$'.

3.3 Characterisation of approximation and head-normalisation

In this section, we will prove two results. First it will be proved that, for every M, B and σ such that $B \vdash M:\sigma$, there is an $A \in \mathcal{A}(M)$ such that $B \vdash A:\sigma$. From this result, the well-known characterisation of (head-)normalisation of lambda terms using intersection types follows easily, i.e. terms, all terms having a (head) normal form are typeable in '$\vdash$' (with a type without ω-occurrences). The second result is the the well-known characterisation of strong normalisation of typeable lambda terms, i.e. all terms, typeable in '$\vdash$' without the type-constant ω, are strongly normalisable.

Using Theorem *2.12*, as for the BCD-system and the strict system, the relation between types assignable to a lambda term and those assignable to its approximants can be formulated as follows:

Theorem 3.12 $B \vdash M:\sigma \iff \exists\, A \in \mathcal{A}(M)\,[B \vdash A:\sigma\,]$.

Proof: $\Rightarrow$) Let $\mathcal{D} :: B \vdash M:\sigma$, then, by Theorem *2.12*, $SN(\mathcal{D})$. Let $\mathcal{D}_0 :: B \vdash N:\sigma$ be the normal form of $\mathcal{D}$ with respect to $\to_{\mathcal{D}}$, then by Lemma *2.4*, $M \twoheadrightarrow_\beta N$, and by Lemma *3.8*, there is $N' \in \lambda\perp$ such that $\mathcal{D}_0 :: B \vdash_\perp N':\sigma$, and $N' \sqsubseteq N$. Since $\mathcal{D}_0$ is a redex-free derivation, $N' \in \mathcal{N}$, so $N' \in \mathcal{A}(M)$. Also, by Lemma *3.7*, $B \vdash N':\sigma$.

$\Leftarrow$) Since $A \in \mathcal{A}(\mathcal{M})$, there is an M' such that $M' =_\beta M$ and $A \sqsubseteq M'$. Then, by Lemma *3.3*, $B \vdash M' : \sigma$, and, by Theorem *1.5 (iii)*, also $B \vdash M : \sigma$. ■

Using this result, the following becomes easy.

Theorem 3.13 ([2]) $\exists B, \sigma\, [B \vdash M : \sigma] \iff$ *M has a head normal form.*

Proof: $\Rightarrow$) If $B \vdash M : \sigma$, then, by Theorem *3.12*, there exists $A \in \mathcal{A}(M)$ such that $B \vdash A : \sigma$. By Definition *3.1*, there exists $M' =_\beta M$ such that $A \sqsubseteq M$. Since $\sigma \in \mathcal{T}_s$, $A \not\equiv \bot$, so A is either $x, \lambda x.A'$ or $xA_1 \cdots A_n$. Since M' matches A, M' is either $x, \lambda x.M_1$ or $xM_1 \cdots M_n$. Then M has a head-normal form.

$\Leftarrow$) If M has a head-normal form, then there exists $M' =_\beta M$ such that M' is either x, $\lambda x.M_1$ or $xM_1 \cdots M_n$, with each $M_i \in \lambda$.

a) $M' \equiv x$. Then $x{:}\varphi \vdash x : \varphi$.

b) $M' \equiv \lambda x.M_1$. Since M_1 is in head-normal form, by induction there is a B such that $B \vdash M_1 : \sigma$. If $x{:}\tau \in B$, then $B \backslash x \vdash \lambda x.M_1 : \tau \to \sigma$, otherwise $B \vdash \lambda x.M_1 : \omega \to \sigma$.

c) $M' \equiv xM_1 \cdots M_n$. Then $x{:}\omega \to \cdots \to \omega \to \varphi \vdash xM_1 \cdots M_n : \varphi$.

Then, by Theorem *1.5 (iii)*, there exists B such that $B \vdash M : \sigma$. ■

In the next section, as in [1] for the strict system, we will prove that the essential intersection type assignment system satisfies the (strong) normalisation properties of the BCD-system.

4 (Strong) normalisation

In this section we will show that, all terms typeable without using the type constant ω in context or conclusion, are normalisable. We will also show that, for the essential notion of type assignment without ω, all terms are strongly normalisable, as first shown in [1]. These results are obtained via the strong normalisation result proved above for derivation reduction.

4.1 Intersection Type Assignment without ω

While building a derivation $B \vdash M : \sigma$ (where ω does not occur in σ and B) for a lambda term M that has a normal form, the type ω is only needed to type sub-terms that will be erased while reducing M to its normal form and that cannot be typed starting from B. We will prove that the set of all terms typeable by the system without ω is the set of all strongly normalisable terms.

Definition 4.1 *i)* $\mathcal{T}_\omega$, the set of *ω-free intersection types*, ranged over by σ, τ etc, is inductively defined by:

$$\sigma \ ::= \ \varphi \mid (\sigma_1 \cap \cdots \cap \sigma_n \to \sigma), \ (n \geq 1)$$

The set of ω-free intersection types is defined by:

$$\{\sigma_1 \cap \cdots \cap \sigma_n \mid n \geq 1 \ \& \ \forall i \in \underline{n} \ [\sigma_i \in \mathcal{T}_\omega]\}$$

ii) On $\mathcal{T}_\omega$ the relation $\leq$ is as defined in Definition *1.1*, except for the second alternative.

$$\begin{array}{rcll} \forall i \in \underline{n} \ [\sigma_1 \cap \cdots \cap \sigma_n & \leq & \sigma_i] & (n \geq 1) \\ \forall i \in \underline{n} \ [\sigma \leq \sigma_i] & \Rightarrow & \sigma \leq \sigma_1 \cap \cdots \cap \sigma_n & (n \geq 1) \\ \sigma \leq \tau \leq \rho & \Rightarrow & \sigma \leq \rho & \end{array}$$

The relations $\leq$ and $\sim$ are extended to bases as before.

iii) If $M : \sigma$ is derivable from a basis B, using only ω-free types and the derivation rules of '$\vdash$', we write $B \vdash_\omega M : \sigma$.

Notice that the only difference between this definition and Definition *1.1* is that $n \geq 1$ rather than $n \geq 0$.

Let '$\vdash_\omega$' denote the notion of derivability obtained from '$\vdash$' by removing the type constant ω. Then the following properties hold:

Lemma 4.2 *i)* $B \vdash_\omega x : \sigma \iff \exists \rho \in \mathcal{T} \ [x{:}\rho \in B \ \& \ \rho \leq \sigma]$.
ii) $B \vdash_\omega MN : \sigma \ \& \ \sigma \in \mathcal{T}_s \iff \exists \tau \in \mathcal{T} \ [B \vdash_\omega M : \tau \to \sigma \ \& \ B \vdash_\omega N : \tau]$.
iii) $B \vdash_\omega \lambda x.M : \sigma \ \& \ \sigma \in \mathcal{T}_s \iff \exists \rho \in \mathcal{T}, \mu \in \mathcal{T}_s \ [\sigma = \rho \to \mu \ \& \ B, x{:}\rho \vdash_\omega M : \mu]$.
iv) $B \vdash_\omega M : \sigma \ \& \ B' \leq B \ \Rightarrow \ B' \vdash_\omega M : \sigma$.
v) If $\mathcal{D} :: B \vdash_\omega M : \sigma$, *then* $\mathcal{D} :: B \vdash M : \sigma$.

Proof: Easy. ■

Lemma 4.3 ([2]) *If A is $\bot$-free, then there are B, and σ, such that $B \vdash_\omega A : \sigma$.*

Proof: By induction on A.
i) $A \equiv x$. $x{:}\varphi \vdash_\omega x : \varphi$.
ii) $A \equiv \lambda x.A'$. By induction there are B, τ such that $B \vdash_\omega A' : \tau$. If x does not occur in B, take an ω-free $\sigma \in \mathcal{T}_s$; otherwise, there exist $x{:}\sigma \in B$, and σ is ω-free. In any case, $B \backslash x \vdash_\omega \lambda x.A' : \sigma \to \tau$.

iii) $A \equiv xA_1 \cdots A_n$. By induction there are $B_1, \ldots, B_n$ and σ_i $(i \in \underline{n})$ such that for every $i \in \underline{n}$, $B_i \vdash_{\omega} A_i : \sigma_i$. Then $\cap\{B_1, \ldots, B_n, x{:}\sigma_1 \to \cdots \to \sigma_n \to \varphi\} \vdash_{\omega} xA_1 \cdots A_n : \varphi$. ■

Theorem 4.4 ([2]) $\exists B, \sigma\, [B \vdash M : \sigma \;\&\; B, \sigma \text{ } \omega\textit{-free}] \iff M$ *has a normal form.*

Proof: $\Rightarrow$) If $B \vdash M : \sigma$, then, by Theorem *3.12*, $\exists A \in \mathcal{A}(M)\, [B \vdash A : \sigma]$. Since B, σ are ω-free, by Lemma *3.10*, this A is $\bot$-free. By Definition *3.1* there exists $M' =_\beta M$ such that $A \sqsubseteq M'$. Then M' itself is in normal form, so, especially, M has a normal form.

$\Leftarrow$) If M' is the normal form of M, then it is a $\bot$-free approximate normal form. Then by Lemma *4.3* there are B, σ such that $B \vdash_{\omega} M' : \sigma$. Then, by Theorem *1.5 (iii)*, $B \vdash M : \sigma$. ■

4.2 Strong normalisation for Intersection Type Assignment without ω

As was remarked in the beginning of this paper, if we are interested in deriving types without ω occurrences, the type constant ω will only be needed in the '$\vdash$'-system to type sub-terms N of M that will be erased while reducing M. In fact, if there is a type ρ such that $B \vdash_{\omega} N : \rho$, then, even for this N we would not need ω. However, there are lambda terms M that contain a sub-term N that must be typed with ω in $B \vdash M : \sigma$, even if ω does not occur in B and σ. We can even find strongly normalisable lambda terms that contain such a sub-term (see also the remark made after Lemma *4.5*).

The following lemma shows a subject expansion result for the ω-free system.

Lemma 4.5 If $B \vdash_{\omega} M[N/x] : \sigma$ *and* $B \vdash_{\omega} N : \rho$, *then* $B \vdash_{\omega} (\lambda x.M)N : \sigma$.

Proof: We focus on the case that $\sigma \in \mathcal{T}_s$, the case that σ is an intersection is just a generalisation. We can assume that x does not occur in B, and proceed by induction on the structure of M.

$(M \equiv x)$: $B \vdash_{\omega} N : \sigma \;\Rightarrow\; B \vdash_{\omega} (\lambda x.x)N : \sigma$

$(M \equiv y \neq x)$: $B \vdash_{\omega} y : \sigma \;\Rightarrow\; B \vdash_{\omega} \lambda x.y : \rho \to \sigma \;\Rightarrow\; B \vdash_{\omega} (\lambda x.y)N : \sigma$. By α-conversion, we can assume that x does not appear in N.

$(M \equiv \lambda y.M')$: Then $(\lambda y.M')[N/x] \equiv \lambda y.(M'[N/x])$, and $\sigma = \delta{\to}\mu$.

$$
\begin{array}{ll}
B \vdash_{\omega} \lambda y.(M'[N/x]) : \delta{\to}\mu \ \&\ B \vdash_{\omega} N : \rho & \Rightarrow\ (\to I) \\
B, y{:}\delta \vdash_{\omega} M'[N/x] : \mu \ \&\ B \vdash_{\omega} N : \rho & \Rightarrow\ (IH) \\
B, y{:}\delta \vdash_{\omega} (\lambda x.M')N : \mu & \Rightarrow\ (\to E) \\
\exists \tau\ [B, y{:}\delta \vdash_{\omega} \lambda x.M' : \tau{\to}\mu \ \&\ B, y{:}\delta \vdash_{\omega} N : \tau] & \Rightarrow\ (\to I)\ \&\ y \notin fv(N) \\
\exists \tau\ [B, y{:}\delta, x{:}\tau \vdash_{\omega} M' : \mu \ \&\ B \vdash_{\omega} N : \tau] & \Rightarrow\ (\to I) \\
\exists \tau\ [B \vdash_{\omega} \lambda xy.M' : \tau{\to}\delta{\to}\mu \ \&\ B \vdash_{\omega} N : \tau] & \Rightarrow\ (\to E) \\
B \vdash_{\omega} (\lambda xy.M')N : \delta{\to}\mu &
\end{array}
$$

$(M \equiv M_1M_2)$: Then $(M_1M_2)[N/x] \equiv M_1[N/x]M_2[N/x]$.

$$
\begin{array}{ll}
B \vdash_{\omega} M_1[N/x]M_2[N/x] : \sigma \ \&\ B \vdash_{\omega} N : \rho & \Rightarrow\ (\to E) \\
\exists \tau\ [B \vdash_{\omega} M_1[N/x] : \tau{\to}\sigma \ \&\ B \vdash_{\omega} M_2[N/x] : \tau] \ \&\ B \vdash_{\omega} N : \rho & \Rightarrow\ (IH) \\
\exists \tau\ [B \vdash_{\omega} (\lambda x.M_1)N : \tau{\to}\sigma \ \&\ B \vdash_{\omega} (\lambda x.M_2)N : \tau] & \Rightarrow\ (\to E)\ \&\ (\to I) \\
\exists \rho_1, \rho_2, \tau\ [B, x{:}\rho_i \vdash_{\omega} M_1 : \tau{\to}\sigma \ \&\ B \vdash_{\omega} N : \rho_1 \ \&\ B, x{:}\rho_2 \vdash_{\omega} M_2 : \tau \ \&\ B \vdash_{\omega} N : \rho_2] & \\
\qquad\qquad \Rightarrow\ (\cap I)\ \&\ (\textit{4.2 (iv)}) & \\
\exists \rho_1, \rho_2\ [B, x{:}\rho_1{\cap}\rho_2 \vdash_{\omega} M_1M_2 : \sigma \ \&\ B \vdash_{\omega} N : \rho_1{\cap}\rho_2] & \Rightarrow\ (\to I) \\
\exists \rho_1, \rho_2\ [B \vdash_{\omega} \lambda x.(M_1M_2) : \rho_1{\cap}\rho_2{\to}\sigma \ \&\ B \vdash_{\omega} N : \rho_1{\cap}\rho_2] & \Rightarrow\ (\to E) \\
B \vdash_{\omega} (\lambda x.(M_1M_2))N : \sigma] & \blacksquare
\end{array}
$$

This result extends by induction (easily) to all contexts: if $B \vdash_{\omega} \mathrm{C}[M[N/x]] : \sigma$ and $B \vdash_{\omega} N : \rho$, then $B \vdash_{\omega} \mathrm{C}[(\lambda x.M)N] : \sigma$.

Notice that the condition $B \vdash_{\omega} N : \rho$ in the formulation of the lemma is essential. As a counter example, take the two lambda terms $\lambda yz.(\lambda b.z)(yz)$ and $\lambda yz.z$. Notice that the first strongly reduces to the latter. We know that

$$z{:}\sigma, y{:}\tau \vdash_{\omega} z : \sigma$$

but it is impossible to give a derivation for $(\lambda b.z)(yz) : \sigma$ from the same basis without using ω. This is caused by the fact that we can only type $(\lambda b.z)(yz)$ in the system without ω from a basis in which the predicate for y is an arrow type. We can, for example, derive

$$B \vdash_{\omega} z{:}\sigma, y{:}\sigma{\to}\tau : (\lambda b.z)(yz)\sigma.$$

We can therefore only state that we can derive

$$\vdash_{\omega} \lambda yz.(\lambda b.z)(yz) : (\sigma{\to}\tau){\to}\sigma{\to}\sigma \text{ and } \vdash_{\omega} \lambda yz.z : \tau{\to}\sigma{\to}\sigma$$

but that we are not able to give a derivation without ω for the statement

$$\vdash_{\scriptscriptstyle\cap} \lambda yz.(\lambda b.z)(yz) : \tau \to \sigma \to \sigma.$$

So the type assignment without ω is not closed for β-equality, but of course this is not imperative. We only want to be able to derive *a* type for each strongly normalisable term, no matter what basis or type is used.

The proof of the crucial lemma as presented below (Lemma *4.7*) and part ($\Leftarrow$) of the proof of Theorem *4.9* are due to Betty Venneri, of the University of Florence, Italy, and goes by induction on the left-most outer-most reduction path.

Definition 4.6 An occurrence of a redex $\mathrm{R} = (\lambda x.P)Q$ in a term M is called the *left-most outer-most redex of* M (*lor*(M)), if:

i) There is no redex R′ in M such that R′ = C[R] (*outer-most*).

ii) There is no redex R′ in M such that $M = \mathrm{C}_0[\mathrm{C}_1[\mathrm{R}']\mathrm{C}_2[\mathrm{R}]]$ (*left-most*).

$M \to_{lor} N$ is used to indicate that M reduces to N by contracting *lor* (M).

The following lemma formulates a subject expansion result for '$\vdash_{\scriptscriptstyle\cap}$' with respect to left-most outer-most reduction. A proof for this property in the context of the strict system appeared in [3]; it is easily modified to fit the essential system.

Lemma 4.7 ([3]) *Let $M \to_{lor} N$, lor$(M) = (\lambda x.P)Q$, $B \vdash_{\scriptscriptstyle\cap} N : \sigma$, and $B' \vdash_{\scriptscriptstyle\cap} Q : \tau$, then there exists B_1, ρ such that $\sigma \leq \rho$, and $B_1 \vdash_{\scriptscriptstyle\cap} M : \rho$.*

We can now show that all strongly normalisable terms are typeable in '$\vdash_{\scriptscriptstyle\cap}$'.

Theorem 4.8 *If M is strongly normalisable, then there are B and σ such that $B \vdash_{\scriptscriptstyle\cap} M : \sigma$.*

Proof: With induction on the maximum of the lengths of reduction sequences for a strongly normalisable term to its normal form (denoted by $\#(M)$).

i) If $\#(M) = 0$, then M is in normal form, and by Lemma *4.3*, there exist B and $\sigma \in \mathcal{T}_s$ such that $B \vdash_{\scriptscriptstyle\cap} M : \sigma$.

ii) If $\#(M) \geq 1$, so M contains a redex, then let $M \to_{lor} N$ by contracting $(\lambda x.P)Q$. Then $\#(N) < \#(M)$, and $\#(Q) < \#(M)$ (since Q is a proper subterm of a redex in M), so by induction $B \vdash_{\scriptscriptstyle\cap} N : \sigma$ and $B' \vdash_{\scriptscriptstyle\cap} Q : \tau$, for some B, B', σ, and τ. Then, by Lemma *4.7*, there exist B_1, ρ such that $B_1 \vdash_{\scriptscriptstyle\cap} M : \rho$. ■

Theorem *4.9* shows that the set of strongly normalisable terms is exactly the set of terms typeable in the intersection system without using the type constant ω.

Theorem 4.9 $\exists B, \sigma\,[B \vdash_{\omega} M:\sigma] \iff M$ *is strongly normalisable with respect to* $\rightarrow_\beta$.

Proof: $\Rightarrow$) If $\mathcal{D} :: B \vdash_{\omega} M:\sigma$, then by Lemma *4.2 (v)*, also $\mathcal{D} :: B \vdash M:\sigma$. Then, by Theorem *2.12*, $\mathcal{D}$ is strongly normalisable with respect to $\rightarrow_{\mathcal{D}}$. Since $\mathcal{D}$ contains no ω, all redexes in M correspond to redexes in $\mathcal{D}$. Since derivation reduction does not introduce ω, also M is strongly normalisable with respect to $\rightarrow_\beta$.

$\Leftarrow$) With induction on the maximum of the lengths of reduction sequences for a strongly normalisable term to its normal form (denoted by $\#(M)$).

a) If $\#(M) = 0$, then M is in normal form, and by Lemma *4.3*, there exist B and $\sigma \in \mathcal{T}_{\omega}$ such that $B \vdash_{\omega} M:\sigma$.

b) If $\#(M) \geq 1$, so M contains a redex, then let $M \rightarrow_{lor} N$ by contracting $(\lambda x.P)Q$. Then $\#(N) \leq \#(M)$, and $\#(Q) \leq \#(M)$ (since Q is a proper subterm of a redex in M), so by induction $B \vdash_{\omega} M:\sigma$ and $B' \vdash_{\omega} Q:\tau$, for some B, B', σ, and τ. Then, by Lemma *4.7*, there exist B_1, ρ such that $B_1 \vdash_{\omega} M:\rho$. ■

5 Alternative type assignment systems

In this section, we will illustrate the result of Section *2.2* by looking briefly at other systems, for which the above results come more easily.

5.1 The relevant type assignment system

It is worthwhile to remark that, in fact, it is the presence of the relation $\leq$ on types, and, especially, the derivation rule (Ax) that greatly complicates the possible solution to the main problem dealt with in this paper. Restricting the setting to a *relevant* system, i.e. where the types assumed for free variables are restricted to those that are *needed* in the derivation, and where $(\rightarrow I)$-rule can only be applied against *used* assumptions over a term-variable, gives a rather straightforward solution.

Below, we will just give the presentation of the $\perp$-variant of the relevant type assignment; the definition of the original system should be clear from that (see [2, 15]).

Definition The $\perp$-variant of *Relevant type assignment* is defined by the following natural deduction system (where all types displayed are strict, except σ in the rules $(\rightarrow I)$, and $(\rightarrow E)$):

$$(Ax)\ \frac{}{x{:}\sigma \vdash^{\perp}_{R} x{:}\sigma}$$

$$(\cap I)\ \frac{B_1 \vdash^{\perp}_{R} M_1{:}\sigma_1 \quad \cdots \quad B_n \vdash^{\perp}_{R} M_n{:}\sigma_n}{\cap\{B_1,\ldots,B_n\} \vdash^{\perp}_{R} M_1 \sqcup \cdots \sqcup M_n{:}\sigma_1 \cap \cdots \cap \sigma_n}\ (n \geq 0)$$

$$(\rightarrow I)\ \frac{B, x{:}\sigma \vdash^{\perp}_{R} M{:}\tau}{B \vdash^{\perp}_{R} \lambda x.M{:}\sigma\rightarrow\tau}\quad \frac{B \vdash^{\perp}_{R} M{:}\tau}{B \vdash^{\perp}_{R} \lambda x.M{:}\omega\rightarrow\tau}\ (x \text{ not in } B)$$

$$(\rightarrow E)\ \frac{B_1 \vdash^{\perp}_{R} M{:}\sigma\rightarrow\tau \quad B_2 \vdash^{\perp}_{R} N{:}\sigma}{\cap\{B_1,B_2\} \vdash^{\perp}_{R} MN{:}\tau}$$

Notice that, by rule $(\cap I)$, $\vdash^{\perp}_{R} \perp{:}\omega$, and that this is the only way in '$\vdash^{\perp}_{R}$' to assign ω to a term.

This notion of type assignment is *relevant* in the sense of [13]: bases contain no more type information than that actually used in the derivation, and, therefore, in the $(\rightarrow I)$-rule, only those types *actually used* in the derivation can be abstracted. This implies that, for example, for the lambda term $(\lambda ab.a)$ types like $\sigma\rightarrow\tau\rightarrow\sigma$ cannot be derived, only types like $\sigma\rightarrow\omega\rightarrow\sigma$. Likewise, for $\lambda x.x$ types like $(\sigma\cap\tau)\rightarrow\sigma$ cannot be derived, only types like $\sigma\rightarrow\sigma$ can.

In this system, contrary to the essential system we considered above, all typeable terms are strongly normalisable with respect to $\rightarrow_{\beta\perp}$; all characterisations results follow from that. We will not discuss the proof for that result in detail here, since it would be very similar to the proof that was given above, or to that in [1]. The main difference lies in the fact that a relevant system is not closed for $\leq$, so in particular no variant of Lemma *2.8* need to be proved. Since the strong normalisation result now talks about terms, also the Computability Predicate can be defined in a less evolved way:

Definition $\mathit{Comp}(B, M, \sigma)$ is recursively defined on σ by:

$$\begin{aligned}
\mathit{Comp}(B,M,\varphi) &\iff B \vdash^{\perp}_{R} M{:}\varphi \ \&\ \mathit{SN}(M)\\
\mathit{Comp}(B,M,\alpha\rightarrow\beta) &\iff (\mathit{Comp}(B',N,\alpha) \Rightarrow \mathit{Comp}(\cap\{B,B'\}, MN, \beta))\\
\mathit{Comp}(\cap\{B_1,\ldots,B_n\}, M_1\sqcup\cdots\sqcup M_n, \sigma_1\cap\cdots\cap\sigma_n)&\\
&\iff \forall i \in \underline{n}\ [\mathit{Comp}(B_i, M_i, \sigma_i)]
\end{aligned}$$

Notice that, by the third part, $\mathit{Comp}(\emptyset, \perp, \omega)$.

Lemma *2.8*, which in turn is essential to prove part $(\leq)$ of the proof of Theorem *2.11*, is not needed here. Instead, the replacement theorem here reads:

Theorem *Let $B = x_1{:}\mu_1, \ldots x_n{:}\mu_n$, $B \vdash M{:}\sigma$, and B' be such that, for every $i \in \underline{n}$, there is a N_i such that Comp(B', N_i, μ_i). Then Comp$(B', M[\overline{N_i/x_i}], \sigma)$.*

For the relevant system this part would be (Ax), and that step of the proof would be trivial. All other proofs follow the line of [1], and are very similar to (simplified versions) of those of Section *2.2*.

5.2 The strict type assignment system [3]

Another system for which the strong normalisation result for derivation reduction comes relatively easy, is the strict system of [1], for which the results proved here were shown in [3]. It can be defined as follows:

Definition The *strict type assignment* [1] is defined by the following natural deduction system (where all types displayed are strict, except σ in the rules $(\rightarrow I)$, and $(\rightarrow E)$):

$$(\cap E): \ \frac{}{B, x{:}\sigma_1 \cap \cdots \cap \sigma_n \vdash_{\rm s} x{:}\sigma_i}\ (n \geq 1, i \in \underline{n}) \qquad (\rightarrow I): \ \frac{B, x{:}\sigma \vdash_{\rm s} M{:}\tau}{B \vdash_{\rm s} \lambda x.M{:}\sigma \rightarrow \tau}$$

$$(\cap I): \ \frac{B \vdash_{\rm s} M{:}\sigma_1 \quad \cdots \quad B \vdash_{\rm s} M{:}\sigma_n}{B \vdash_{\rm s} M{:}\sigma_1 \cap \cdots \cap \sigma_n}\ (n \geq 0) \qquad (\rightarrow E): \ \frac{B \vdash_{\rm s} M{:}\sigma \rightarrow \tau \quad B \vdash_{\rm s} N{:}\sigma}{B \vdash_{\rm s} MN{:}\tau}$$

Notice that, essentially, the difference between the relevant and the strict systems lies in going from derivation rule (*Ax*) to ($\cap$*E*). In fact, derivation rule ($\cap$*E*) is implicitly present in the system '$\vdash_{\rm R}$', since there the intersection of types occurring in bases is produced using the $\bigcap$-operator. The system '$\vdash_{\rm s}$' does not use this operator; instead, it allows for the selection of types from an intersection type occurring in a basis, regardless if all components of that intersection type are useful for the derivation. In this sense, the strict system is not relevant.

A difference between the strict and the essential system is that the selection of types from those provided in the bases is done through the relation $\leq$, not just selecting from an intersection. In the essential system, it is possible to derive $\vdash_{\perp} \lambda x.x{:}(\alpha \rightarrow \beta) \rightarrow (\alpha \cap \gamma) \rightarrow \beta$, which is not possible in '$\vdash_{\rm s}$'.

In this strict system, as for the essential system, it is possible to type non-normalisable terms; however, derivation reduction is strongly normalisable, as could be expected. Again we will omit almost all the proof for that result here, since it would be very similar to the

proof that was given above. In particular, the Computability Predicate can be defined in a similar way.

The main difference between the solution of [3] and the one presented here lies in the fact that here we prove Lemma *2.8*, whereas in [3], it is not needed at all. However, the difference between the strict system and the one considered in this paper, rule $(\cap E)$ versus (Ax), makes that the first part of the Replacement Lemma becomes:

$(\cap E)$: Then $x{:}\sigma_1 \cap \cdots \cap \sigma_n \in B'$, $\sigma = \sigma_i$ for some $i \in \underline{n}$, and $\mathcal{D}_i :: B \vdash N_i : \sigma_i$. By Definition, *Comp* $(\mathcal{D}_i)$, and $\mathcal{D}^0\,[\overrightarrow{\mathcal{D}/x{:}\mu}] = \mathcal{D}_i$.

Acknowledgment

I would like to thank Mariangiola Dezani, who on so many occasions tried, with me, to understand what was going on in the $\perp$-system, and to make the proofs work.

References

[1] S. Bakel. Complete restrictions of the Intersection Type Discipline. *Theoretical Computer Science*, 102(1):135–163, 1992.

[2] S. Bakel. Intersection Type Assignment Systems. *Theoretical Computer Science*, 151(2):385–435, 1995.

[3] S. Bakel. Cut-Elimination in the Strict Intersection Type Assignment System is Strongly Normalising. *Notre Dame journal of Formal Logic*, 45(1), 2004.

[4] S. Bakel and M. Fernández. Strong Normalization of Typeable Rewrite Systems. In Jan Heering, Karl Meinke, Bernhard Möller, and Tobias Nipkow, editors, *Proceedings of HOA'93. First International Workshop on Higher Order Algebra, Logic and Term Rewriting*, Amsterdam, the Netherlands. *Selected Papers*, volume 816 of *Lecture Notes in Computer Science*, pages 20–39. Springer-Verlag, 1994.

[5] S. Bakel and M. Fernández. (Head-)Normalization of Typeable Rewrite Systems. In Jieh Hsiang, editor, *Proceedings of RTA'95. 6th International Conference on Rewriting Techniques and Applications*, Kaiserslautern, Germany, volume 914 of *Lecture Notes in Computer Science*, pages 279–293. Springer-Verlag, 1995.

[6] S. Bakel and M. Fernández. Approximation and Normalization Results for Typeable Term Rewriting Systems. In Gilles Dowek, Jan Heering, Karl Meinke, and Bernhard Möller, editors, *Proceedings of HOA'95. Second International Workshop on Higher Order Algebra, Logic and Term Rewriting*, Paderborn, Germany. *Selected Papers*, volume 1074 of *Lecture Notes in Computer Science*, pages 17–36. Springer-Verlag, 1996.

[7] S. Bakel and M. Fernández. Normalization Results for Typeable Rewrite Systems. *Information and Computation*, 2(133):73–116, 1997.

[8] H. Barendregt. *The Lambda Calculus: its Syntax and Semantics*. North-Holland, Amsterdam, revised edition, 1984.

[9] H. Barendregt, M. Coppo, and M. Dezani-Ciancaglini. A filter lambda model and the completeness of type assignment. *journal of Symbolic Logic*, 48(4):931–940, 1983.

[10] M. Coppo and M. Dezani-Ciancaglini. An Extension of the Basic Functionality Theory for the λ-Calculus. *Notre Dame journal of Formal Logic*, 21(4):685–693, 1980.

[11] M. Coppo, M. Dezani-Ciancaglini, and M. Zacchi. Type Theories, Normal Forms and D_∞-Lambda-Models. *Information and Computation*, 72(2):85–116, 1987.

[12] H.B. Curry and R. Feys. *Combinatory Logic*, volume 1. North-Holland, Amsterdam, 1958.

[13] F. Damiani and P. Giannini. A Decidable Intersection Type System based on Relevance. In M. Hagiya and J.C. Mitchell, editors, *Proceedings of TACS'94. International Symposium on Theoretical Aspects of Computer Software,* Sendai, Japan, volume 789 of *Lecture Notes in Computer Science*, pages 707–725. Springer-Verlag, 1994.

[14] M. Dezani-Ciancaglini and I. Margaria. A characterisation of F-complete type assignments. *Theoretical Computer Science*, 45:121–157, 1986.

[15] P. Gardner. Discovering Needed Reductions Using Type Theory. In M. Hagiya and J.C. Mitchell, editors, *Proceedings of TACS'94. International Symposium on Theoretical Aspects of Computer Software,* Sendai, Japan, volume 789 of *Lecture Notes in Computer Science*, pages 555–597. Springer-Verlag, 1994.

[16] J.-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1989.

[17] J-L. Krivine. *Lambda-Calcul – Types et Modèles*. Etudes et Recherches en Informatique. Masson, Paris, 1990.

[18] C. Retoré. A note on intersection types. INRIA Rapport de recherche 2431, INRIA, France, 1994.

[19] S. Ronchi Della Rocca. Principal type scheme and unification for intersection type discipline. *Theoretical Computer Science*, 59:181–209, 1988.

[20] S. Ronchi Della Rocca and B. Venneri. Principal type schemes for an extended type theory. *Theoretical Computer Science*, 28:151–169, 1984.

[21] W.W. Tait. Intensional interpretation of functionals of finite type I. *journal of Symbolic Logic*, 32(2):198–223, 1967.

[22] C.P. Wadsworth. The relation between computational and denotational properties for Scott's D_∞-models of the lambda-calculus. *SIAM J. Comput.*, 5:488–521, 1976.

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399